

TOBIAS TRATNER

00803337

Master's Thesis

Master's degree programme: Technology Management, M.Eng.

Faculty NUW. Applied Natural Sciences and Industrial Engineering

Deggendorf Institute of Technology

Graz University of Technology

Implementation of Time Series Data based Digital Time Studies for Manual Processes within the Context of a Learning Factory

Supervisor

Dipl.-Ing. Maria Hulla

Institute of Innovation and Industrial Management, TU Graz

Evaluator

Prof. Dr.-Ing. Peter Firsching

Graz, May 2021

Affidavit

Name of the student: Tobias Tratner

Name of the evaluator: Prof. Dr.-Ing. Peter Firsching

Topic of the master's thesis:

"Implementation of Time Series Data based Digital Time Studies for Manual Processes within the Context of a Learning Factory"

1. "I assure, that I wrote the thesis myself, that I have not submitted it to any other means for examination purposes, that I have given all sources and resources used and that I have marked literal and analogous quotations."

Date

Signature

2. "I agree that the master thesis, I have prepared, can be made accessible to a wider public by the library of the Deggendorf Institute of Technology."

Yes

No

Date

Signature

Abstract

The steadily advancing globalization significantly shapes today's business environment for companies. Therefore, companies are increasingly under immense cost pressure and need to improve their production efficiency and product quality to remain competitive. Industry 4.0 applications that result from the advancing digitization offer great potential for long-term cost savings. Implementing time studies for mechanical activities can identify potential for improvement in the production process and enable them to be rectified. This thesis is concerned with combining the subject areas Industry 4.0 and the implementation of manual time studies. For this purpose, a digital time recording of manual activities was implemented in the LEAD Factory, the learning factory of the Institute of Innovation and Industrial Management at Graz University of Technology. Therefore, a mobile sensor kit and an IoT platform, provided by the factorycube of Industrial Analytics, were used. Using sensors, existing data from an RFID system, and an energy monitoring system, all activities on a selected workstation in the LEAD Factory can be documented and analyzed. This automated time recording enables long-term measurements to analyze working times and possible anomalies. The collected data is stored in so-called time-series databases and processed using various methods. The data is displayed on a dashboard using a visualization program. One focus of the work was the design of the data processing architecture with two different time-series data models, as well as the conception and development of methods for data processing in the context of time studies. A relational and a NoSQL database system were used equally. The use of two very different approaches should show the possibilities of both systems and enable an assessment of the two systems. Based on a utility analysis, both approaches are evaluated and compared using selected criteria. Thus, a clear recommendation can be made for one of the two approaches. Making the results of the work available to an open-source community, they can be used as a basis for the implementation of similar applications. In addition, the work shows through a digital time recording the huge potential to improve productivity in case of using existing data in a production environment.

Abstract

Die stetig fortschreitende Globalisierung prägt das heutige Geschäftsumfeld für Unternehmen erheblich. Unternehmen stehen daher zunehmend unter einem enormen Kostendruck und müssen ihre Produktionseffizienz und Produktqualität stetig verbessern, um wettbewerbsfähig zu bleiben. Industrie 4.0-Anwendungen, die sich aus der fortschreitenden Digitalisierung ergeben, bieten ein großes Potenzial für langfristige Kosteneinsparungen. Durch die Implementierung von Zeitstudien für mechanische Aktivitäten können Verbesserungspotenziale im Produktionsprozess identifiziert und umgesetzt werden. Diese Arbeit befasst sich mit der Kombination und Verknüpfung der Themenbereiche Industrie 4.0 und der Durchführung manueller Zeitstudien. Zu diesem Zweck wurde in der LEAD Factory, der Lernfabrik des Instituts für Innovation und Industriemanagement der Technischen Universität Graz, eine digitale Zeiterfassung manueller Aktivitäten implementiert. Hierfür wurden ein mobiles Sensorkit und eine IoT-Plattform verwendet, die vom Factorycube von Industrial Analytics bereitgestellt wurden. Mithilfe von Sensoren, vorhandenen Daten aus einem RFID-System und einem Energieüberwachungssystem können alle Aktivitäten auf einer ausgewählten Workstation in der LEAD Factory dokumentiert und analysiert werden. Diese automatisierte Zeiterfassung ermöglicht Langzeitmessungen zur Analyse von Arbeitszeiten und möglichen Anomalien. Die gesammelten Daten werden in sogenannten Zeitreihendatenbanken gespeichert und mit verschiedenen Methoden verarbeitet. Die Daten werden mithilfe eines Visualisierungsprogramms auf einem Dashboard angezeigt. Ein Schwerpunkt der Arbeit war das Design der Datenverarbeitungsarchitektur mit zwei verschiedenen Zeitreihendatenmodellen sowie die Konzeption und Entwicklung von Methoden zur Datenverarbeitung im Rahmen von Zeitstudien. Ein relationales und ein NoSQL-Datenbanksystem wurden gleichermaßen verwendet. Die Verwendung von zwei sehr unterschiedlichen Ansätzen sollte die Möglichkeiten beider Systeme aufzeigen und eine Bewertung der beiden Systeme ermöglichen. Basierend auf einer Nutzwertanalyse werden beide Ansätze anhand ausgewählter Kriterien bewertet und miteinander verglichen. Somit kann eine klare Empfehlung für einen der beiden Ansätze abgegeben werden. Indem die Ergebnisse der Arbeit einer Open-Source-Community zur Verfügung gestellt werden, können sie als Grundlage für die Implementierung ähnlicher Anwendungen verwendet werden. Darüber hinaus zeigt die Arbeit anhand einer digitalen Zeiterfassung das enorme Potenzial zur Verbesserung der Produktivität bei Verwendung vorhandener Daten in einer Produktionsumgebung.

Contents

| | | |
|-------|---|----|
| 1 | Introduction | 1 |
| 1.1 | Motivation and initial situation | 1 |
| 1.2 | Goals and research questions | 2 |
| 1.3 | Challenges including the LEAD Factory..... | 3 |
| 1.4 | Structure of this thesis | 6 |
| 2 | Literature review..... | 7 |
| 2.1 | Development of trainings in a learning factory..... | 7 |
| 2.2 | Internet of Things | 10 |
| 2.3 | Big data..... | 12 |
| 2.4 | Radio frequency identification..... | 12 |
| 2.5 | Basics of databases..... | 13 |
| 2.5.1 | Relational databases..... | 14 |
| 2.5.2 | NoSQL-databases | 16 |
| 2.5.3 | Time-series databases | 18 |
| 2.6 | Time-series database systems..... | 20 |
| 2.6.1 | InfluxDB and Flux | 22 |
| 2.6.2 | TimescaleDB and SQL | 25 |
| 2.7 | Data visualization..... | 28 |
| 2.8 | Data analytics | 31 |
| 2.7.1 | Component model..... | 32 |
| 2.7.2 | Selected methods | 33 |
| 2.9 | Time structure analysis | 35 |
| 2.10 | Summary | 38 |
| 3 | Methods..... | 39 |
| 3.1 | Experimental setup | 39 |
| 3.1.1 | Factorycube | 41 |
| 3.1.2 | IO-Link sensors..... | 42 |
| 3.1.3 | LEAD-Factory sensors..... | 43 |
| 3.2 | Preparation of time recording | 44 |
| 3.3 | Software architecture design | 50 |

| | | |
|-------|--|-----|
| 3.3.1 | Software stack 1 | 50 |
| 3.3.2 | Software stack 2 | 53 |
| 3.4 | Summary | 55 |
| 4 | Results..... | 56 |
| 4.1 | Implementation with software-stack 1 | 56 |
| 4.1.1 | Implementation of data collection | 56 |
| 4.1.2 | Implementation of data processing..... | 58 |
| 4.2 | Implementation with software-stack 2 | 66 |
| 4.2.1 | Implementation of data collection | 66 |
| 4.2.2 | Implementation of data processing..... | 67 |
| 4.3 | Implementation of the visualization | 71 |
| 4.4 | Comparison of the two solutions with different software-stacks..... | 74 |
| 4.5 | Conception of training and teaching opportunities | 81 |
| 4.6 | Summary | 83 |
| 5 | Discussion..... | 84 |
| 5.1 | Limitations and related software update..... | 84 |
| 5.2 | Answering the research questions..... | 85 |
| 5.1.1 | Research question 1 | 85 |
| 5.1.2 | Research question 2 | 86 |
| 5.1.3 | Research question 3 | 87 |
| 6 | Conclusion | 88 |
| 6.1 | Summary | 88 |
| 6.2 | Outlook..... | 90 |
| | Bibliography | 92 |
| | Appendix..... | 98 |
| | Appendix A: Work Instruction | 98 |
| | Appendix B: Software Stack 1 (Influxdata)..... | 110 |
| | Appendix C: Software Stack 2 (Node-red and TimescaleDB) | 123 |
| | Appendix D: Survey on the weighting of the comparison criteria | 127 |

List of figures

| | |
|--|----|
| Figure 1. Setup of the LEAD Factory in the full optimized digital state..... | 4 |
| Figure 2. Exploded view of the scooter, assembled in the LEAD Factory | 5 |
| Figure 3. Learning pyramid or cone of learning (based on [11]) | 7 |
| Figure 4. Build-up of empirical knowledge in learning factories (based on [15]) | 8 |
| Figure 5. Levels of the learning factory design (based on [17])..... | 9 |
| Figure 6. Learning factory curriculum guide [18] | 9 |
| Figure 7. Possible Internet of Things devices [24]..... | 10 |
| Figure 8. Schematic functionality of an RFID system (based on [32]) | 13 |
| Figure 9. Architecture and components of a database system [7]..... | 14 |
| Figure 10. Schematic representation of relational databases (based on [35])..... | 15 |
| Figure 11. Example table in a relational database [7]..... | 15 |
| Figure 12. Schematic representation of NoSQL databases (based on [35]) | 17 |
| Figure 13. Different NoSQL databases [7]..... | 17 |
| Figure 14. Database popularity trend [45] | 19 |
| Figure 15. Comparison on influxdata and Timescale (based on [47]) | 21 |
| Figure 16. Popularity ranking among TSDBs (based on [45]) | 21 |
| Figure 17. Schematic overview Influx [41] | 23 |
| Figure 18. Tagset data model of Influx [47] | 23 |
| Figure 19. Relational data model for time-series data [47] | 26 |
| Figure 20. Schematic representation of a hypertable and the belonging chunks [47]..... | 26 |
| Figure 21. Example table and results table (based on [7])..... | 27 |
| Figure 22. Visualization is an essential step in data analysis (based on [54])..... | 28 |
| Figure 23. Illustration of perception using the attribute color (based on [54]) | 28 |
| Figure 24. Exemplary representation of the possible visualizations of data in a dashboard panel | 30 |
| Figure 25. Phases of data analysis projects (based on [58]) | 31 |
| Figure 26. REFA standard time recording method (based on [5]) | 37 |
| Figure 27. Basic setup of the factorycube | 40 |

| | |
|--|-----|
| Figure 28. Used IO-Link Sensors..... | 42 |
| Figure 29. Manufacturing process of the digital state in the LEAD Factory [72] | 44 |
| Figure 30. Workstation 4 of the LEAD Factory | 45 |
| Figure 31. Installation of sensors at the workstation | 46 |
| Figure 32. Installation of sensors at the workstation | 47 |
| Figure 33. Relevant architecture of the software stack..... | 53 |
| Figure 34. Data processing architecture in node-red for software stack 2 | 54 |
| Figure 35. Configuration of the telegraf agent | 57 |
| Figure 36. Token for the influx telegraf (shown via the putty client) | 58 |
| Figure 37. Example table to position sensor data processing | 60 |
| Figure 38. Example data of the vibration sensor | 61 |
| Figure 39. Tool container with tool separation and sensor holder | 62 |
| Figure 40. Measurement curve with two different areas..... | 63 |
| Figure 41. Representation of the grouping in different states with multiple values | 66 |
| Figure 42. Showing all signals via the MQTT Explorer | 67 |
| Figure 43. Main overview of activity tracking on the dashboard | 71 |
| Figure 44. Longterm scooter production data..... | 72 |
| Figure 45. Display of defective components and their distribution | 73 |
| Figure 46. Display of the individual sensor activities | 73 |
| Figure 47. Definition of a Grafana interface (left) and Implementation of a graph data source (right) with Influx | 74 |
| Figure 48. Collaboration with ia: as part of the private beta via GitHub | 85 |
| Figure 49. Finished setup in the LEAD-Factory..... | 89 |
| Figure 50. Task 1 downsampling step 1 | 110 |
| Figure 51. Task 2-1 processing position sensor | 111 |
| Figure 52. Task 2-2 processing position sensor | 112 |
| Figure 53. Task 3 processing vibration sensor | 113 |
| Figure 54. Task 4 processing induction sensor..... | 114 |
| Figure 55. Task 5 processing PMD profiler..... | 115 |
| Figure 56. Task 2, 3, 4, 5 downsampling | 116 |

| | |
|--|-----|
| Figure 57. Task 6 to 13 join/union tables..... | 117 |
| Figure 58. Task 14 reading and processing of the LEAD Factory data..... | 118 |
| Figure 59. Task 15 processing longterm data 1..... | 119 |
| Figure 60. Task 15 processing longterm data 2..... | 120 |
| Figure 61. Task 15 processing longterm data 3..... | 121 |
| Figure 62. Task 15 substep processing longterm data 4 | 122 |
| Figure 63. Main data processing implemented in node-red..... | 123 |
| Figure 64. Data processing PMD profiler..... | 123 |
| Figure 65. Data processing induction sensor | 123 |
| Figure 66. Data processing vibration sensor | 124 |
| Figure 67. Data processing position sensor | 124 |
| Figure 68. Reading and processing of the LEAD Factory data | 124 |
| Figure 69. Data processing vibration sensor original | 125 |
| Figure 70. Data processing vibration sensor surrounding LEAD Factory | 125 |
| Figure 71. Merging of the processed data | 125 |
| Figure 72. Downsampling of the processed data | 125 |
| Figure 73. Processing longterm data 1 | 126 |
| Figure 74. Processing longterm data 2 | 126 |

List of tables

| | |
|--|----|
| Table 1. IoT Reference Model (based on [24] and [25]) | 11 |
| Table 2. Advantages and disadvantages of relational databases [37]..... | 16 |
| Table 3. Advantages and disadvantages of NoSQL databases [39]..... | 18 |
| Table 4. Measuring points along the work process | 48 |
| Table 5. Overview of all activities, their assigned tags, sensors, and connections.... | 49 |
| Table 6. Presentation of the evaluated comparison criteria..... | 75 |
| Table 7. Matrix for pair comparison method completed by the author..... | 76 |
| Table 8. Weighting of the criteria after interviewing 11 participants..... | 76 |
| Table 9. Utility analysis for the two factorycube software stacks | 80 |
| Table 10. Didactical transformation of sensor training with the factorycube..... | 82 |

List of abbreviations

| | |
|--------|--|
| IIM | Institute of Innovation and Industrial Management |
| LEAD | Lean, Energy efficiency, Agility, and Digitalization |
| IA: | Industrial Analytics |
| IoT | Internet of Things |
| SQL | Structured query language |
| NoSQL | Not only structured query language |
| RFID | Radio-frequency identification |
| DB | Database |
| DBMS | Database management system |
| TSDB | Time series database |
| TSDBMS | Time series database management system |
| MA | Moving average |
| EMA | Exponential moving average |
| EA | Exponential average |
| MEA | Moving exponential average |
| BPMN | Business Process Model and Notation |
| MQTT | Message Queuing Telemetry Transport |
| UMH | United Manufacturing Hub |

1 Introduction

This chapter gives an overview of this master thesis. In the beginning, the motivation for creating this work is presented. For this purpose, an overview of the initial situation is also given. Furthermore, the research goals and research questions are presented, which will be answered in the course of this thesis. This works' main challenges are explained to show which problems are to face in this context. To be able to understand the following chapters, a short introduction to the LEAD Factory of the Institute of Innovation and Industrial Management (IIM) at the Graz University of Technology is given. Finally, in this chapter's last subsection, the thesis structure will be explained.

1.1 Motivation and initial situation

Today's business environment for companies is significantly shaped by the steadily advancing globalization and its positive and negative effects. Companies no longer only compete on a national level. On top of that, they have to assert themselves against international competition. *"Globalization drives the development towards better, that means here: results that are more marketable and cheaper, but also of higher quality and more and more perfectly tailored to the interests of individual customer groups."* [1] Companies are therefore increasingly under immense cost pressure and are required to take all necessary measures to increase their competitiveness to be able to save costs in the long term [2]. In high-wage countries, the focus on possible cost savings is primarily on personnel and process efficiency. These areas offer great potential for sustainable growth through the application of Industry 4.0, given by the advancing digitization of production environments [3, 4]. In this field, this thesis deals with providing a digitized and automated application for the detection of process weaknesses or optimization potential in manual activities to be able to increase personnel and process efficiency. For this purpose, actual times of manual activities are automatically recorded over a long period by using a digital retrofitting solution in the field of the Internet of Things (IoT), whereby the monitored process is digitally mapped. Based on the obtained data in such a long-term observation as part of a time structure analysis, measures can be taken to increase personnel and process efficiency. Besides, this time data can be used for other applications in the product and production environment, without determining it manually [5]. The time recording of manual activities through a digital retrofit, which is carried out in this work, will be taking place in the learning factory of the Institute of Innovation and Industrial Management at the Graz University of Technology. In a production facility close to industrial reality, a scooter that consists of 60 parts is build on several workstations. One of these workstations is being retrofitted with a mobile IoT sensor kit and a time series database platform to carry out a time study. Currently, only the total time of a work step at the work station can be determined in the learning

Introduction

factory, but not the associated sub-steps. With the implementation of the digital retrofit, provided by the factorycube of Industrial Analytics, the whole work step can be broken down to a time structure analysis. This offers the possibility to make optimization potential visible in individual process steps and take suitable measures [6]. The focus here is on using IoT sensors and a time-series database (TSDB), which is necessary for storing and processing so-called Big data and the associated analysis methods around time-series data. Time-series databases can use two different data models, so two different database systems are used for implementation in this thesis [7]. In this way, it can finally be assessed which data model is more suitable for performing a time structure analysis.

Through the implementation in a learning factory, students and employees of companies can also be introduced to such digitization measures as part of training courses in order to counteract a lack of qualified personnel in the industry and enable companies to use digitization potential [8].

1.2 Goals and research questions

The required tasks to solve the described problems can be differentiated into two main subject blocks, which are using the factorycube infrastructure. Topic one area includes the generation and evaluation of time series data in the context of IoT to carry out a time structure analysis for manual activities within the LEAD Factory framework as an opportunity to identify optimization potential. This generation and evaluation of time series data will be done with two different TSDBs, based on a relational and non-relational data model. Furthermore, it should show the possibilities of time series databases combined with IoT devices. The last topic deals with the conception of training possibilities created by using the factorycube.

As part of this master thesis, the work offers various possible research questions. However, due to the time frame limitation for the master thesis, not all of them can be considered. The chosen research questions, which are essential to the author, should be checked and answered during this thesis. The following questions were therefore selected for a closer examination.

Research Question 1 (chapter 3 and 4)

- Which software stack, including data model (relational or NoSQL) for storing data in time-series databases, is most suitable for performing a time study analysis for manual activities, and how is this reflected in the practical implementation using the two different databases TimescaleDB and InfluxDB?

Research Question 2 (chapters 4.1 and 4.2)

- Can existing data from the LEAD Factory be used, and which methods in data processing must be developed to perform a time study analysis with time-series data?

Research Question 3 (chapter 4.5)

- Can the training of participants by using a factorycube represent a suitable way for imparting knowledge regarding digitization in the field of IoT time-series data analytics?

1.3 Challenges including the LEAD Factory

The main challenges to face in this thesis can be broken down into the same two subject blocks, as mentioned in the previous chapter. To be able to carry out a time structure analysis by using the infrastructure of the factorycube initially includes the preparation of the time recording and the definition of requirements for suitable concepts and methods for time series data analytics. Therefore, a Business Process Model and Notation (BPMN) is a convenient way to visualize the necessary architecture. To store and process the large amount of data created by IoT sensors, a time-series database is required. The digital time study analysis will be taking place on a manual work process at a workstation of the LEAD Factory. Therefore, suitable methods must be conceptualized to receive processed data, which can be evaluated. This methodology will be the base for implementing the data processing, which will be done in two different time-series databases, InfluxDB2.0 and TimescaleDB. So, two different data models will be used, i.e., the methods must be adapted to the respective model and the query language. The visualization of the data will then be done with Grafana. Therefore, several interfaces have to be implemented. Finally, the collected data can be evaluated and assessed to carry out a time study analysis.

The creation of training in connection with the factorycube requires a realistic scenario in which the participants can do their first programming tasks independently. Furthermore, they will be introduced to the generation, processing, and evaluation of time series data in the context of IoT digitization applications.

LEAD Factory

In order to impart knowledge in the areas of Lean management, Energy efficiency, Agile production, and Digitization to students and partners from the industry, the **LEAD** Factory, as can be seen in Figure 1, was set up at the IIM Institute of the Graz University of Technology.

Introduction



Figure 1. Setup of the LEAD Factory in the full optimized digital state

In this downsized industrial production facility, equipped with an assembly line, the workshop and training participants can use the hands-on learning environment to collect valuable knowledge to design efficient production environments. In this production environment, the participants assemble the TU Graz scooter (Figure 2), which consists of 60 parts.

During the LEAD Factory workshops, the participants start in a non-optimized state (1). Inefficient workflows with no leveled workload, unclear assembly instructions, and insufficient tools characterize this state primarily. The participants should be made aware of the potential for improvements. Due to recorded key figures and an analysis of the production, possible improvements are worked out by the participants and practically implemented step by step. The participants are provided with the necessary knowledge in 5S, flow principle, Heijunka, Kanban, seven types of waste, and value stream mapping by the trained specialist staff of the IIM Institute [9]. This approach aims to reorganize the assembly process to an optimized production line called lean state (2) to raise the whole process's efficiency [10]. In a further setup, the digitalized state (3), the existing lean state is digitized through the use of various technologies and assistance systems and thus further optimized. The use of an RFID system for tracking the production progress, a smart Andon system for the automatic monitoring of assembly times, a camera system for detecting workpiece positions at workplaces, and recording gestures and facial expressions to control work instructions can be cited. The LEAD Factory is continuously expanded with new applications to bring companies and their staff closer to a wide range

Introduction

of digitization measures, benefits, and implementation costs. So, participants can be introduced to digitization measures and trained in their use [8]. Through the playful learned experience, the participants are motivated to use this knowledge profitably in their companies.



Figure 2. Exploded view of the scooter, assembled in the LEAD Factory

Further, the LEAD Factory serves as a research platform for digitization options in production, in which applications can be tested in a small, realistic environment. The time series data-based time structure analysis described in this thesis was conceptualized and implemented in this context.

1.4 Structure of this thesis

The following chapter (2) is dedicated to the literature review. This chapter's primary function is to build a funded knowledge base in all topics necessary for this thesis' work. In addition, it should show that there is a research gap, which is addressed by this thesis.

Chapter 3 introduces the factorycube and its different software stacks. Besides, the used sensors and the setup in the LEAD Factory will be examined. In the first step, the preparation for performing a time recording will be done, and the requirements for implementing a digital time recording system are defined. Finally, this chapter presents the required system architectures in a BPMN for both software stacks.

The implementation of the two different approaches concerning the two different software stacks is explained in Chapter 4. This mainly includes collecting and processing the data to make them usable for the participants of the learning factory. For this purpose, the visualization of the obtained data should be explained. Furthermore, the two implemented time analysis software solutions based on different software stacks will be compared. Finally, a possible teaching concept will be given.

In chapter 5, the results are discussed by answering the research questions with the help of the gained knowledge. Furthermore, the used system's limitations and the related software update are explained.

In the last chapter (6), an outlook on possible improvements and extensions that could be built on top of this thesis is being given. Finally, the whole thesis is summarized in the conclusion.

2 Literature review

This chapter aims to discuss the necessary fundamentals to understand this thesis. Since the digital time structure analysis implementation takes place in the LEAD factory, the concept of training in learning factories is explained in more detail. In addition, an insight into the topic of the IoT, which covers this work, will be given. As part of IoT, the basics of radio-frequency identification (RFID) systems and database types will be examined in more detail. The time-series database solutions from InfluxDB and TimescaleDB will be explained as special databases with the associated query languages Flux and SQL (Structured Query Language). The Grafana tool used to visualize data will also be discussed in more detail. Overall, data processing plays a significant role in this thesis, so insight into the topic of time series data analysis is necessary. Finally, the basics of time recording analysis will be introduced to the reader.

2.1 Development of trainings in a learning factory

“Tell me, and I forget. Teach me, and I may remember. Involve me, and I learn.”

(Confucius, 551–479 BCE, Chinese philosopher)

This quote from the Chinese philosopher Confucius describes in a fitting way how differently effective learning can be. The core of this quote is reflected in the learning pyramid of the National Training Laboratory Institute for Applied Behavioral Science, according to Figure 3.

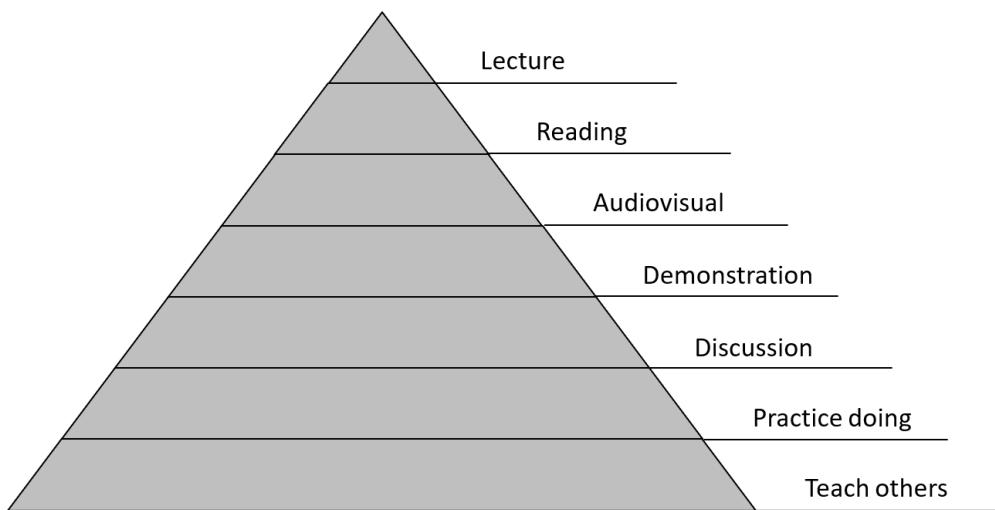


Figure 3. Learning pyramid or cone of learning (based on [11])

The pyramid describes how to absorb knowledge optimally and how to keep it in our memory. Typically, the learning pyramid is shown with percentages for the individual learning methods. However, these are controversial as there is no scientifically founded background for these percentages [12, 13]. For this reason, percentages are not given here.

According to the learning pyramid, learning by listening is the least efficient. Minimal knowledge is retained and can be reproduced. The learning efficiency increases considerably by choosing the right type of learning. Teaching other people is seen as the most effective method. In order to be able to teach others, a high degree of self-understanding of the respective topic is required. The practical application is located one level below this learning method [14]. At this level, learning factories are located.

“Vocational education and training aim to develop professional skills that enable employees to act self-organized and responsibly in open, unsafe, and complex situations. The development of theoretical and empirical knowledge is a requirement for their development. Through the possibility of experimenting and collecting specific experiences, learning factories serve to build up empirical knowledge and enable a formalized form of experience-based learning.” [15] Figure 4 shows the way of transforming formal learning into empirical knowledge.

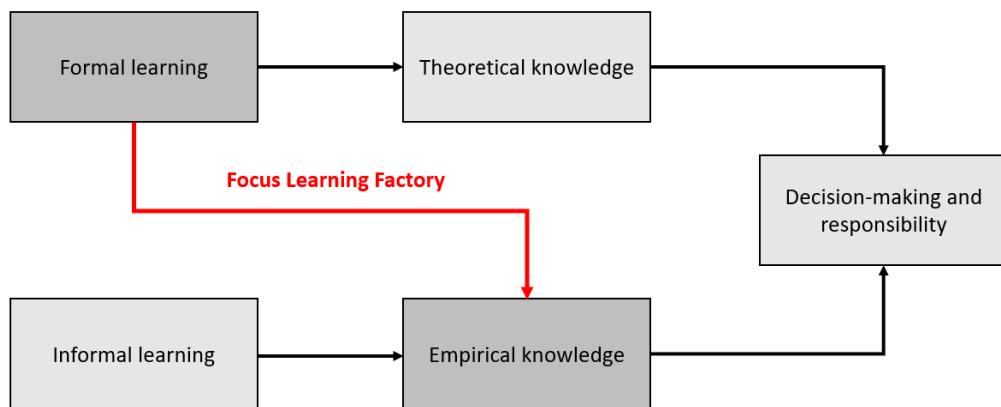


Figure 4. Build-up of empirical knowledge in learning factories (based on [15])

With this form of “game-based learning,” the combination of content with playful elements offers the opportunity to apply acquired theoretical knowledge immediately in the real-life environment of a learning factory. Besides, this learning type should focus on the participants’ autonomy, goal setting, collaboration, communication, and reflection [16]. In summary, a learning factory “stands for a didactically and methodically based teaching and learning environment that ideally covers the entire production process and adjacent divisions. In the present understanding, a learning factory is an “off-the-job” teaching-learning Environment, in which learners can consciously experiment and experience possible consequences and mistakes.” [15]

Technical experts often carry out the development and conception of learning factories, whereby mostly didactical approaches are not considered. As a result, users’ gain in competence can be much more inefficient. A competency-oriented method was developed to prevent this problem in which a learning factory is divided into three design levels. As shown in Figure 5, these are macro-level, meso-level, and micro-level. The macro-level describes the environment of the learning factory as well as the production and the basics

Literature review

of the desired learning process. The individual teaching modules are defined on the meso-level below. The imparted competencies and the required teaching-learning sequences for this are considered. In the lowest level, the micro-level, the various teaching-learning situations are finally described and defined. Therefore, a learning factory can have several teaching modules containing several learning situations [17].

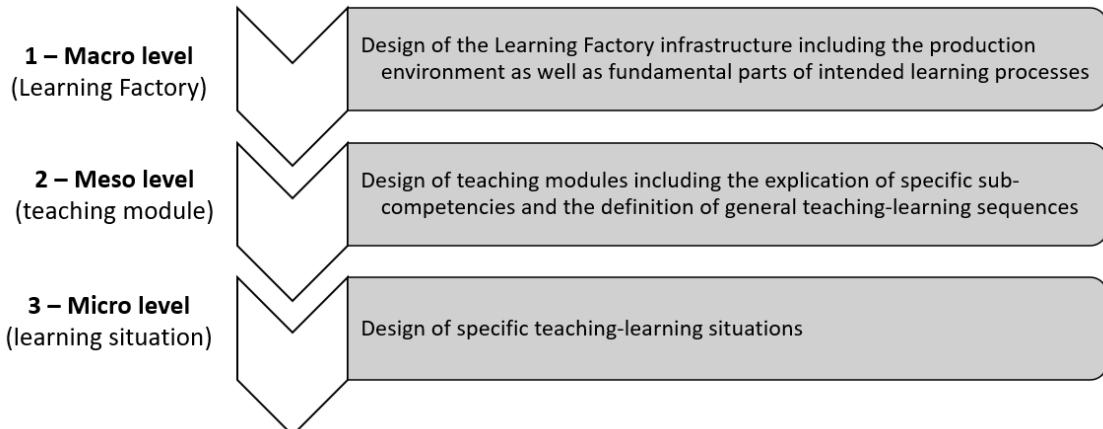


Figure 5. Levels of the learning factory design (based on [17])

As shown in Figure 6, two didactic transformations are carried out in each of the mentioned levels. In the first transformation, the learning and competence goals are defined from various requirements. These requirements concerning the content, personal and organizational aspects. This transformation's relevant question would be: "*What are relevant learning targets and contents for involved stakeholders?*" [17] The second didactic transformation describes the change of the defined learning objectives into a concrete learning factory concept based on the question "*How can those learning targets and the content be addressed in the learning factory?*" [17].

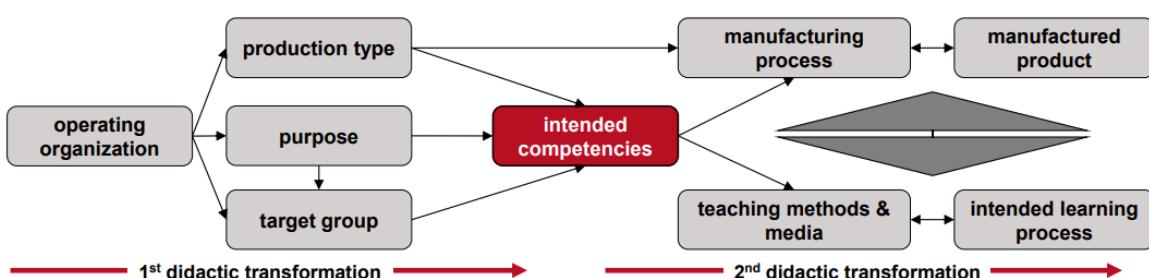


Figure 6. Learning factory curriculum guide [18]

Another important aspect of learning factories is their function as a research platform for developing innovations, such as new prototypes, product technologies, production technologies, and processes. In this environment, innovations can be researched realistically without negatively affecting companies' value creation. Furthermore, the costs for developments can be reduced through the use of learning factories [19]. Due to this fact, a learning factory is an excellent choice for the implementation of this master's thesis.

2.2 Internet of Things

As part of Industry 4.0¹, the Internet of Things is continually finding its way into our everyday lives. IoT describes the merge of the virtual and the physical world into what is known as cyber-physical systems [20]. Cyber-physical systems are defined as “*transformative technologies for managing interconnected systems between its physical assets and computational capabilities*” [21].

Whereas the definition of IoT is not clearly defined. It essentially describes the networking of everyday objects with an Internet-like structure. As a result, various devices used in everyday life can be assigned to the Internet of Things, such as smartphones, TV, or even a surveillance camera [22]. Multiple devices that can be connected to the Internet of Things are shown in Figure 7. Nevertheless, the term “IoT” is still used for devices “*that were not supposed to be connected to the internet in the first place.*” [23] The German Bundestag’s official definition describes IoT instead as “*the technical vision to integrate objects of all kinds into a universal digital network.*” [22]



Figure 7. Possible Internet of Things devices [24]

An IoT system consists of various technologies and applications, which can be divided into different levels or layers named as Multi-Layer-Modell. For this purpose, the IoT Reference Model was created by CISCO, as shown in Table 1. The IoT Reference Model contains seven levels, which describe the data flow from data acquisition to application [25]. For the implementation of the master thesis in the Internet of Things environment, levels one to six are particularly relevant, the so-called edge-side layer and server or cloud-side layer. In addition, at a later point in the master thesis, the existing components of the factorycube are divided into the various levels.

¹ “Industry 4.0 utilizing the power of communications technology and innovative inventions to boost the development of the manufacturing industry.” [75]

Literature review

Table 1. IoT Reference Model (based on [25] and [26])

| Level | Explanation | Technology | Layer type |
|-------------------------------------|--|----------------------------------|----------------------------|
| 7. Collaboration and Processes | Involving People and Business Processes | API management | user-side layer |
| 6. Application | Reporting, Analytics, Control | App technologies | Server or cloud-side layer |
| 5. Data Abstraction | Processing, Summary, and Aggregation of data | Big Data, AI and Cloud Computing | |
| 4. Data Accumulation | Storage of a large amount of data | Big Data and Cloud Computing | |
| 3. Edge Computing ² | Local Data Element Analysis and Transformation | Edge Computing | edge-side layer |
| 2. Connectivity | Communication and Processing Units | Network technologies | |
| 1. Physical Devices and Controllers | Data acquisition (“Things”) | Sensors and actuators | |

The first level, the so-called “edge level,” contains the “things,” devices such as sensors or machines that generate data (sensors) or use data (actuators). Level 2 establishes the connection between level 1 and level 2. At the connectivity level, network technologies such as the fifth generation of mobile communications 5G, wireless, or the communication protocol Ethernet ensure that devices can communicate with the storage level. This creates a flow of data along with the levels. The obtained data can then be preprocessed locally at the edge computing level. Edge computing is used to process data locally on a production machine, for example, before the data is transferred to a cloud. This level does not necessarily have to be present. Rather it represents an option depending on the application. All the information is stored and managed on the storage level with the help of big data technologies and cloud computing. These technologies are necessary in order to be able to handle the enormous flood of IoT data. The data abstraction level forms the heart of data analytics in IoT systems. At this point, all data from various sources are brought together and will be processed and prepared suitably, using technologies such as big data (see chapter 2.3), artificial intelligence called AI, and cloud computing. The generated and processed data in the previous levels is displayed and visualized properly at the application level. A possibility for that can be the visualization on so-called dashboards [25, 26].

² Edge computing is the decentralized data processing at the edge of the network. This can be done, for example, directly on site on a production machine [37].

2.3 Big data

In the course of Industry 4.0 and the Internet of Things, another term is widespread: “Big data.” This involves large amounts of data that can no longer be managed with conventional methods. Usually, it is unstructured data from a wide variety of data sources. This can be text, 3D graphics, photos, or even language. The variety is almost unlimited. The Gartner Group describes the term Big data as follows: *“Big data is high-volume, high-velocity, and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”* [27] In detail, Big data applications, can be described with the 4 V’s [28].

- Volume (volume of data, starts with terabyte)
- Variety (variety of stored data)
- Velocity (speed in the processing of data)
- Value (increase in company value using Big data)

Big data will continue to be closely linked to development in the future. It will lead to significant demands on database systems and their storage, management, and processing. Not only the requirements for database systems are increasing, but also the requirements for companies to use big data applications profitably. An analysis by the MIT Sloan Management Review and the IBM Institute for Business Value showed that successful companies use Big Data applications around five times more than less successful companies. This fact should be a motivation for every company to get ready for the future with the help of Big data [29].

2.4 Radio frequency identification

The RFID technology was already developed during the Second World War in radar and radio research. A long-range transponder system was used to identify friend or foe in combat aircraft. RFID belongs to Auto-ID-Systems, which means it is a system for automatic identification. With increasing development in integrated circuits, RFID tags could be reduced to a single integrated circuit and could thus find their way into everyday life [30]. Today, RFID technology is used in various areas of everyday life. Movement tracking is a key application. Due to their small size, RFID tags can easily be attached to moving objects or carried by living beings. A typical example of this is tracking pets or their use in supply chain management to track products across the entire supply chain [23]. Another focus of use is security, e.g., in combination with credit cards as payment technology or ID cards to enter buildings. There are many exciting applications, and their development will continue in the future [31].

The essential components of an RFID system are, as shown in Figure 8, an RFID transponder, an RFID reader, and an information system. The RFID transponder is a passive component, i.e., it does not have its own energy supply. The transponder consists of a microchip and an integrated antenna. The RFID reader also has an antenna and thus forms a transmission field with a limited range. If a transponder penetrates the described transmission field, it is activated by the transmission field's energy. Data and the tact are transmitted to the transponder. The transponder can then send response data back to the reader, which forwards the data to an information system. This is a so-called bidirectional connection [32].

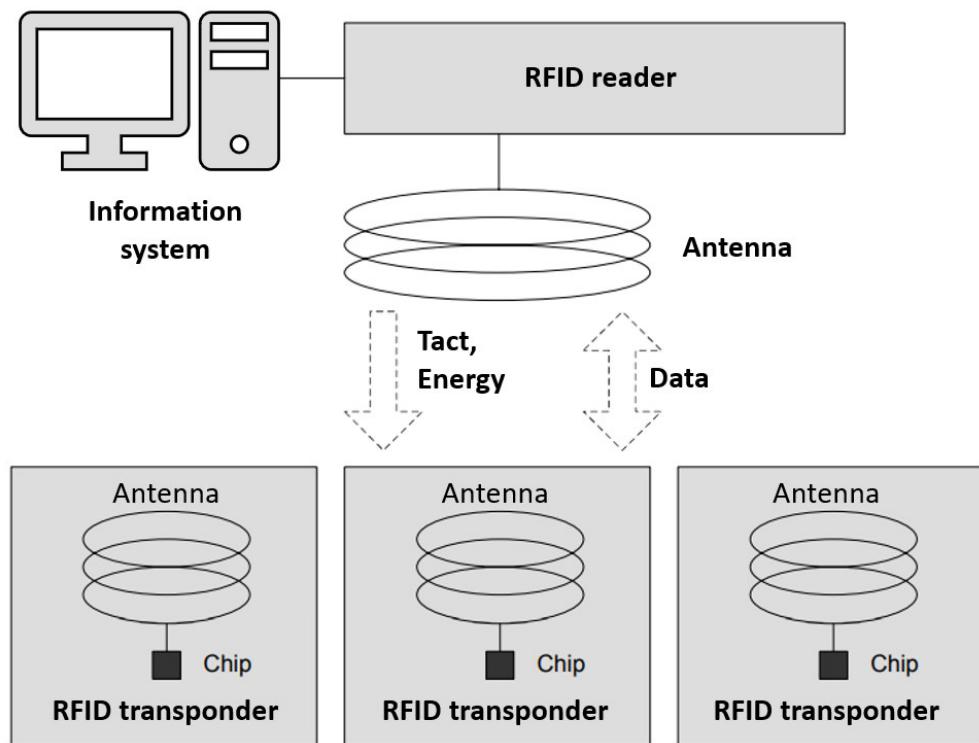


Figure 8. Schematic functionality of an RFID system (based on [32])

2.5 Basics of databases

The term “database” or “database system” has become an integral part of today’s life. The US software and hardware manufacturer Oracle defines the term database as follows: *“A database is an organized collection of structured information or data, that is typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS³). Together, the database and the DBMS and the associated applications are referred to as the database system, which is often limited to the pure database.”* [33]

³ A DBMS is software that defines the model of a database system and is therefore the decisive component in order to be able to set up, manage and use a database [78].

Figure 9 shows the basic architecture and components of a database system. The system's task can be defined by describing, storing, and querying data. The database system consists of a storage component and a management component. The storage component consists of the database for the structured storage of the data itself and a method and knowledge base. The latter two provide essential database technologies for managing and evaluating the data. So these methods are available and do not have to be redeveloped at great expense. Additional communication between the database system and other networks enables information to be exchanged with other sources.

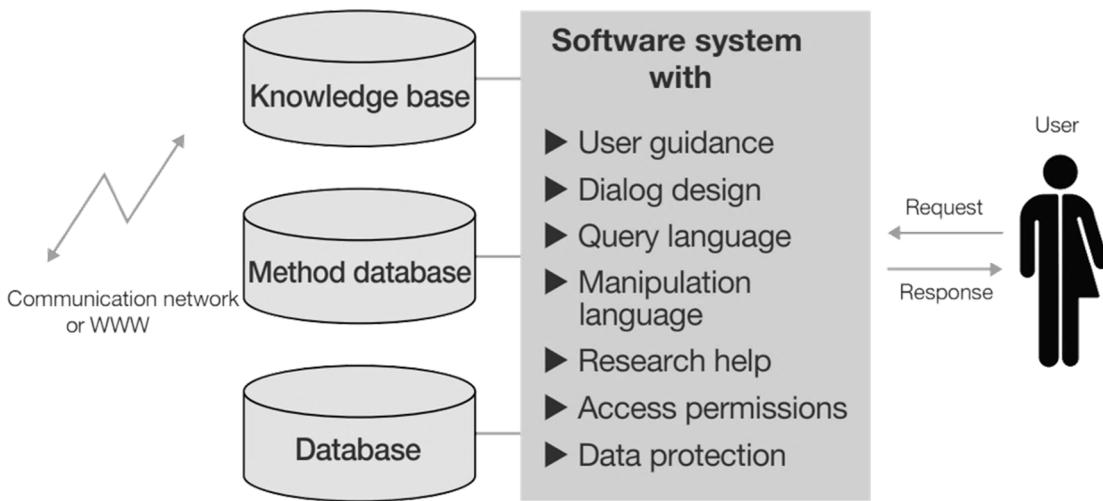


Figure 9. Architecture and components of a database system [7]

The software system serves as an interface between the storage component and the user. The software system interacts with the user and offers the possibility of querying and manipulating the data through a query and manipulation language. The system can also provide the user with important aids in his work. Besides, the system defines user access rights and thus data protection measures. In this chapter, the database component is considered in more detail [7].

There are a large number of different database types that can be selected depending on the desired type of application. The most common database types are relational databases, object-oriented databases, distributed databases, data warehouses, NoSQL databases, and diagram databases. This chapter introduces the two most essential database types, relational and NoSQL databases [33, 28].

2.5.1 Relational databases

The most important subgroup of databases are relational databases, also known as SQL databases. This type of database has existed since 1970 and was first developed by E. F. Codd based on Codd's relational model [34]. The term "SQL" stands for "Structured Query Language," which is today known as the international standard language for data queries in such systems. Besides, it enables data descriptions, data manipulation, and data selection.

Layout and function

According to Figure 10, a relational database system is an integrated system for managing data, consisting of a tabular form and an SQL-based management system for user interaction [35].

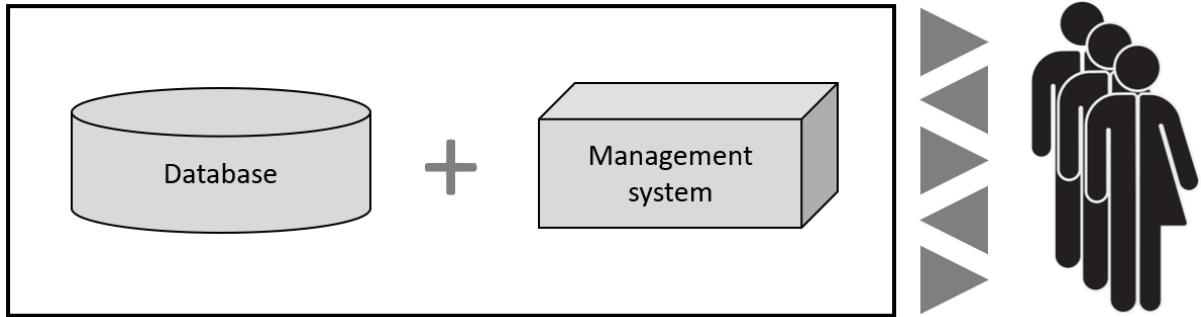


Figure 10. Schematic representation of relational databases (based on [35])

A relational database is characterized by various properties, including the **model**, **schema**, **language**, **architecture**, **multi-user operation**, **consistency assurance**, **data security**, and **data protection**. The **model** describes the relational model. The data is accordingly stored in tables. These tables consist of data records, so-called tuples described by attributes, and a unique key. Figure 11 shows an example table for different employees in a company. [36] The definition of these attributes and keys is stored in a database **schema**. The key attribute uniquely identifies each tuple, consisting of a combination of the record attributes. As already mentioned, the standard language SQL is used as the query **language**. The **database system's architecture** should guarantee data independence so that changes can be carried out if necessary, without impairing applications dependent on the database. Another essential characteristic is that a large number of users can use the system without hindering themselves or influencing data. This fact is known as a **multi-user operation**.

The diagram shows two representations of a relational database table. On the left, a simplified diagram of the 'EMPLOYEE' table is shown with columns labeled 'E#', 'Name', and 'City'. Arrows point from the table name, an attribute ('Name'), and the primary key ('E#') to their respective parts. On the right, a more detailed version of the same table is shown with four rows of data. The columns are labeled 'E#', 'Name', and 'City'. The rows are labeled E19, E4, E1, and E7. Arrows point from the table name, a column header ('Name'), a data value ('Howard'), and a row label ('E1') to their corresponding labels. A bracket at the bottom right is labeled 'Record (row or tuple)'.

| E# | Name | City |
|-----|---------|-----------|
| E19 | Stewart | Stow |
| E4 | Bell | Kent |
| E1 | Murphy | Kent |
| E7 | Howard | Cleveland |

Figure 11. Example table in a relational database [7]

The data in relational databases is structured data, so it is necessary to ensure consistency and maintain data integrity. Furthermore, **data security** and **data protection** are becoming more and more important and critical. Therefore, this applies above all to databases and is an important criterion for database systems. Table 2 shows the advantages and disadvantages of this type of database.

Table 2. Advantages and disadvantages of relational databases [37]

| Advantages | Disadvantages |
|--|--|
| <ul style="list-style-type: none"> ➤ Simple data model ➤ Low data redundancy ➤ High data consistency ➤ Quantity-oriented data processing ➤ Uniform query language (SQL) | <ul style="list-style-type: none"> ➤ Rigid scheme ➤ No hierarchical database schema ➤ Segmentation of data ➤ Worse performance compared to NoSQL-databases |

Use cases

Today relational databases are among the most widespread database systems in companies and organizations. The internationally established SQL standard greatly simplifies the use of such databases. An SQL database is usually the first choice if a database is required. However, if this reaches its limits due to its disadvantages, NoSQL databases could be a common alternative. In general, it can be said that relational databases are best suited for structured data with low or medium data volumes [28].

2.5.2 NoSQL-databases

With the increasing importance of IoT and Big data, more powerful and larger database systems are required. Both properties are limited in relational databases. Besides, the variety of different data has increased sharply in recent years. These facts make it necessary to use other database systems that meet the growing requirements in this area.

Layout and function

NoSQL databases are databases that do not use a relational approach to enable the storage and processing of large amounts of data. These are mainly characterized by data storage in different formats, i.e., not rational and other database languages. NoSQL is usually translated as “Not only SQL” [28]. Figure 12 illustrates the widely distributed, decentralized data storage architecture, with the individual data being stored depending on the type of database. The best-known types of NoSQL databases are key-value

databases, document-oriented databases, column-oriented databases, and graph databases.

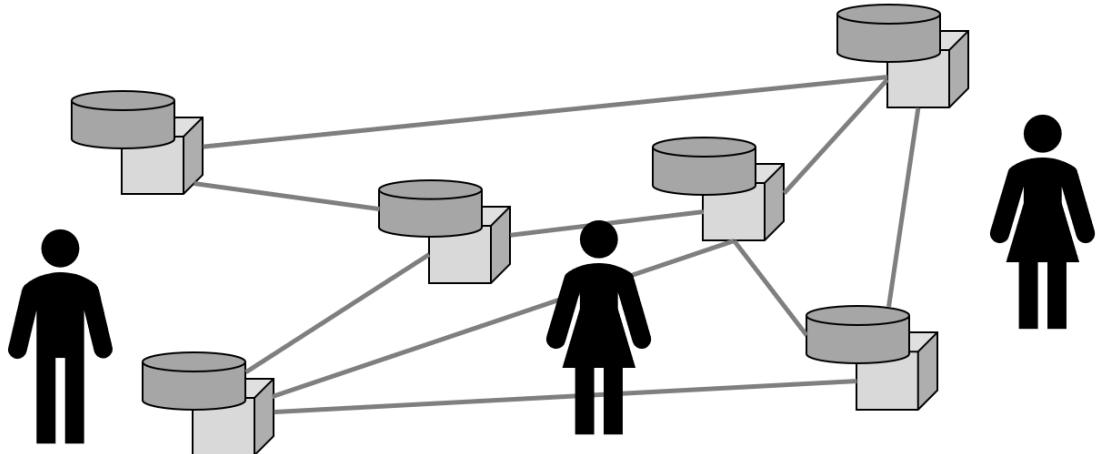


Figure 12. Schematic representation of NoSQL databases (based on [35])

In Figure 13, three of the mentioned databases are shown as examples. In reality, they are selected as required. Because of the decentralized structure, not only different data types can be processed, almost any number of additional data records can also be added. Furthermore, the data records can be processed decentrally and thus in time. NoSQL databases are therefore ideally suited for big data [7].

Describing NoSQL databases in the same way as relational databases, the aspects **model**, **architecture**, **3 V**, **schema**, **multi-user operation**, and **guarantee of consistency** must be addressed.

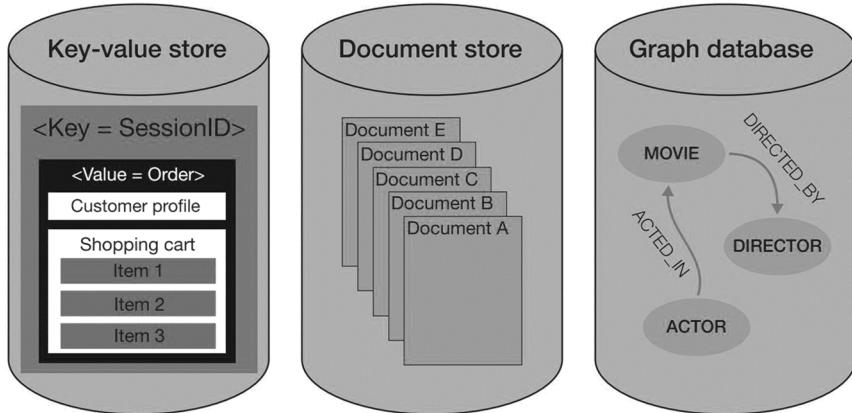


Figure 13. Different NoSQL databases [7]

As already described, the database model is non-relational, characterized by a decentralized architecture. Accordingly, no fixed database schema is used. This construction positively influences data volume, data variety, and real-time processing, called velocity. Interaction by several users is also supported. Only the consistency of the data can vary [35, 38]. Table 3 shows the advantages and disadvantages of NoSQL databases.

Table 3. Advantages and disadvantages of NoSQL databases [39]

| Advantages | Disadvantages |
|---|--|
| <ul style="list-style-type: none"> ➤ Elastic scalability ➤ Support of large amounts of data ➤ Dynamic schemes (various types of data can be stored) ➤ High system performance (data queries and manipulation) ➤ Auto-sharing (data can be distributed to multiple servers) | <ul style="list-style-type: none"> ➤ Worse data consistency ➤ No standardized query language |

Use cases

Due to NoSQL databases' advantages, they are particularly suitable for applications with a high volume of unstructured data and short response times. Therefore, they can be found primarily in e-commerce, social media, and smart devices, in which large amounts of data are generated. The database system's unlimited horizontal scalings are particularly advantageous here [7]. Due to the large variety of NoSQL databases and their applications, a large number of database applications have established themselves in the market. These must be selected as required. In contrast to relational databases, there is no all-round database solution.

2.5.3 Time-series databases

In order to explain the basics of time series databases, a definition of time series data is required. Ted Dunning and Ellen Friedman describe time series data in their fundamental work on time series databases as follows: “*The basis of a time series is the repeated measurement of parameters over time together with the times at which the measurements were made. Time series often consists of measurements made at regular intervals, but the regularity of time intervals between measurements is not a requirement.*” [40] Data is streamed in real-time and is usually never updated or revised. This leads to the advantage that the data history can also be traced back later, and possible problems can be identified. This enables possible forecasting with the help of time series. Any issues or gaps in the time structure can be caused by errors in the time stamp assignment, e.g., an incorrect system time or sensors' failure [25]. With time-series data, one axis always represents the time [41]. Thus, time-series form an antipole to cross-sectional data, as they are based on “*temporally disordered observations for different feature carriers that are often collected at a uniform point in time.*” [42] Panel data form a hybrid form of these two types of data. They include a combination of the properties of both data types. Time series data is mainly known from the area of financial markets, where they depict

Literature review

market events in real-time on charts. Today, time-series data is increasingly finding its way into IoT and Industry 4.0 applications and contributing to big data generation. Large amounts of sensor-based time-series information are generated, especially in industrial production. These are collected at regular intervals and given a timestamp [43]. Besides, there are also many time series data applications, such as anomaly detection or predictive analytics. But time series data also play an increasingly important role in business intelligence tools or monitoring systems. This list could go on and on. Time series data will continue to increase in the future and will play a more significant role [44]. As shown in Figure 14, this trend is also reflected in the growing popularity of time series databases. DB-Engines regularly publishes a ranking based on various methods.

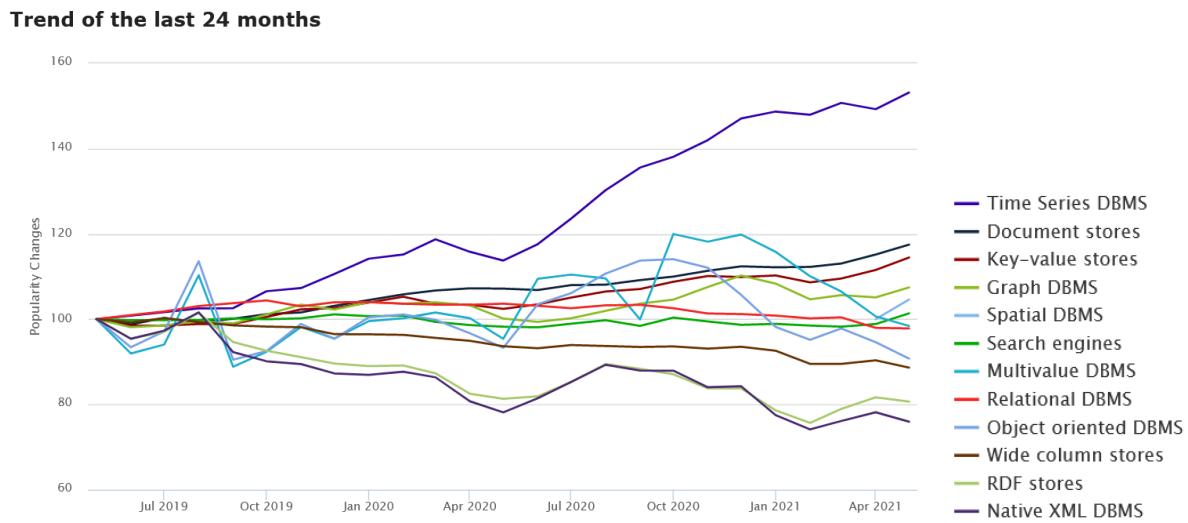


Figure 14. Database popularity trend [45]

Time-series databases are the fastest-growing type of databases over the past two years. Time series have a unique structure. Regularly, a name is assigned to the time series, which enables an exact assignment. A time series consists of many time-series data, i.e., lines comparable to the tuples in relational databases. Each bar is characterized by a timestamp and can therefore be assigned in terms of time. The timestamp format differs depending on the desired operating mode and application but is usually in the accuracy range of milliseconds. “2019-09-18T12: 00: 00.000000000” represents an exemplary timestamp. In addition to the timestamp, any values and tags can be assigned. In most cases, these contain the actual measured values, i.e., the information carriers and tags that specify the measuring points’ origin more precisely. In the case of a sensor system, a line can be characterized, for example, by the measured value, indication of the measuring system, an indication of the sensor, an indication of the table, and further information. The time difference, the time span between two adjacent timestamps, is called the time range - the smallest possible time range as granularity. The granularity is a significant time-series indicator, as it is decisive for selecting a suitable time-series database system. The lower the desired granularity, the more efficient a database system must be. This already starts with data storage. Large amounts of data must be stored in

a shorter time. The same applies to the processing of data. The higher data density requires the processing of more and more data. Therefore, the granularity must always be considered in connection with the desired data analysis since the scope of the analysis activity also ties up a database system's performance resources. Therefore, the combination of granularity and the analysis scope is essential concerning the required performance of a database system [44].

Database systems are required to store and process these time series. Such systems consist of the database as a storage component, and it also includes an associated management system in the form of suitable software. Four properties essentially characterize a Time-series database (TSDB) “*a DBMS that can (i) store a record that consists of a timestamp, value, and optional tags, (ii) store multiple records grouped together, (iii) can query for records and (iv) can contain time ranges in a query is called TSDB.*” [44] A TSDB should provide essential functions for executing queries and the associated data analysis. In addition, the change of the original granularity, e.g., for downsampling, should be supported.

Due to many different application scenarios and the associated large number of significantly different requirements, a large number of TSDS have established themselves in the market [45]. There are essentially three different classes of time series databases, TSDB based on relational databases (SQL), TSDB based on NoSQL databases, and TSDB as an independent solution. A general evaluation of the various systems is hard to make since the systems' properties are very individually distributed and must be considered depending on the application. In principle, however, the trend towards NoSQL TSDB can be recognized, as this is the more suitable solution for vast amounts of data and the query based on time. Furthermore, the standard stand-alone solutions are based on a non-relational model and can be referred to as a NoSQL database [46].

2.6 Time-series database systems

In the work belonging to this master thesis, two different time-series databases are used. These are the independent time-series database InfluxDB from influxdata and its query language Flux and the partially independent database TimescaleDB, which uses the standard query language SQL. Thus, two completely different approaches of time-series databases are used in this work, as Figure 15 shows. Influxdata uses a non-relational data model, the tagset data model, which will be explained under 2.6.1, whereas Timescale uses a relational data model. In addition, in a case with Flux, NoSQL is used, and in the case of Timescale, SQL is used as the query language.



Figure 15. Comparison on influxdata and Timescale (based on [47])

The IT portal DB-engine regularly conducts surveys in the area of database popularity. The evaluation is based on criteria such as “*number of quotations on websites, interest among users, discussion rate, employment offers based on the system and number of professional profiles.*” [45, 48] It should be noticed that this ranking gives no indication of which database offers the physically better system, only the current popularity. As Figure 16 shows, Influx leads the ranking by a large margin. Furthermore, TimescaleDB is steadily increasing in the ranking.

| Rank | | | DBMS | Database Model | Score | | |
|----------|----------|----------|--------------|--------------------------|----------|----------|----------|
| May 2021 | Apr 2021 | May 2020 | | | May 2021 | Apr 2021 | May 2020 |
| 1. | 1. | 1. | InfluxDB | Time Series, Multi-model | 27.17 | +0.62 | +6.25 |
| 2. | 2. | 2. | Kdb+ | Time Series, Multi-model | 8.26 | +0.41 | +2.89 |
| 3. | 3. | 3. | Prometheus | Time Series | 5.76 | +0.03 | +1.45 |
| 4. | 4. | 4. | Graphite | Time Series | 4.56 | +0.03 | +1.10 |
| 5. | 5. | ↑ 8. | TimescaleDB | Time Series, Multi-model | 2.90 | +0.13 | +1.13 |
| 6. | ↑ 7. | 6. | Apache Druid | Multi-model | 2.67 | +0.04 | +0.76 |
| 7. | ↓ 6. | ↓ 5. | RRDtool | Time Series | 2.46 | -0.25 | -0.06 |
| 8. | 8. | ↓ 7. | OpenTSDB | Time Series | 1.80 | +0.05 | -0.01 |
| 9. | 9. | 9. | Fauna | Multi-model | 1.49 | -0.03 | +0.54 |
| 10. | 10. | ↑ 11. | GridDB | Time Series, Multi-model | 1.03 | +0.05 | +0.59 |

Figure 16. Popularity ranking among TSDBs (based on [45])

The following is an overview of the two different databases’ structures and functionalities. Besides, the two previously mentioned query languages are highlighted. Specific details, especially regarding practical work, are deepened in chapters 3 and 4.

2.6.1 InfluxDB and Flux

“InfluxDB is the essential time-series toolkit — dashboards, queries, tasks, and agents all in one place.”, this is how influxdata describes their database [41]. The presented information relates to version 2.0, with which the query language Flux was introduced, and system components were further linked. Particular attention must be paid to this in literature research since the information on previous versions only applies to version 2.0 to a very limited extent due to the significant changes mentioned above. In general, it can be stated that the data situation regarding InfluxDB v2.0 and its introduction in November 2020 is still very limited.

InfluxDB is an open-source, schemaless time-series database management system designed as a stand-alone solution. It was first introduced in 2013. The unique feature is that no other DBMS is therefore required. It is programmed in the language Go and works on Linux or OS X. Because of the open-source architecture, it supports nearly every common programming language like python, java, and many more. If the term “Influx” is used in the following, this means the entire DBMS with its individual components. Influx is mainly used in the following areas: Application Performance Monitoring, Industrial IoT, DevOps monitoring, and real-time analytics. Wherever high volumes of data arise or a summary of data over a significant period is required [45].

Architecture

As shown in Figure 17, the system essentially consists of two components: the InfluxDB database and the Telegraf-agent. In the previous versions, the system consisted of the so-called TICK-stack, T - Telegraf, I - InfluxDB, C - Chronograph, and K - Kapacitor. The ICK components were combined in the latest version into a single binary in InfluxDB 2.0. This enables a significantly simplified use and a higher system performance [49]. The Telegraf agent collects data from various sources. There are now over 200 different Telegraf plugins for collecting data because of the open-source architecture. This data can come from sensor systems, third-party APIs (application programming interface), other databases, clouds, and many more. Due to this variety of plugins, various data sources can be used. The Telegraf agent collects this data and writes it to any Influx database destination. The data is transferred to a suitable format and is fitted with a time stamp. All these parameters are set in the configuration of the Telegraf agent [49].

Literature review

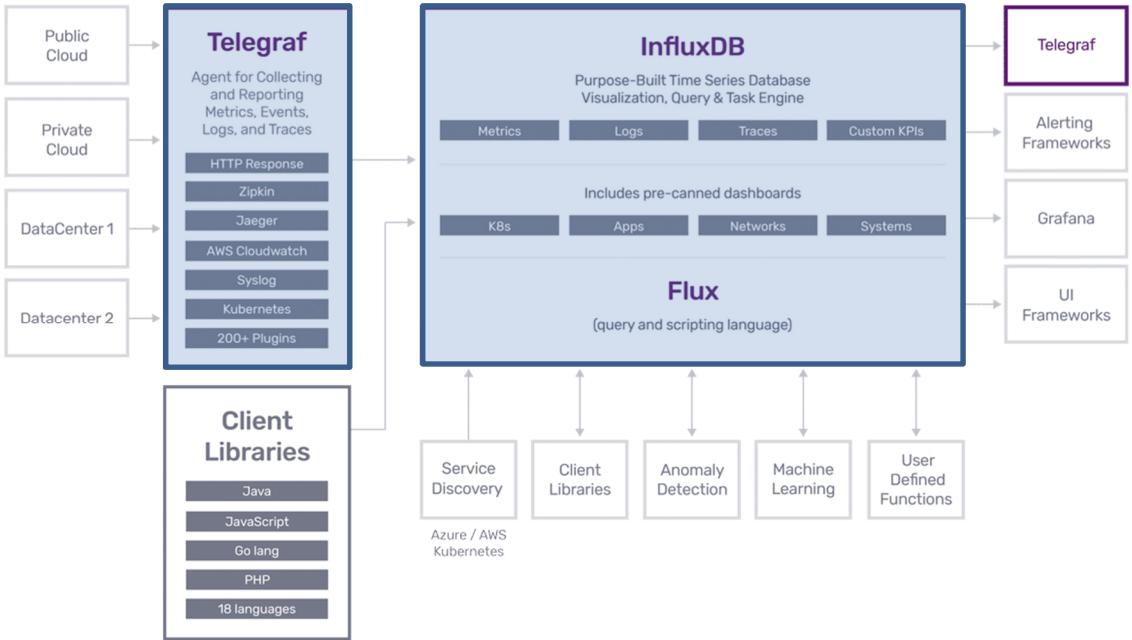


Figure 17. Schematic overview Influx [41]

Data model

InfluxDB is the heart of the DBMS. It fulfills several tasks, such as storing or retaining data/metrics, queries, and visualization. The data collected by a single telegraf is stored in so-called buckets, of which any number can exist. It must always be precisely defined which data is stored in which bucket or is extracted. The data is stored according to a protocol for data ingestion, called “line protocol” in the format: <measurement-name>, <tag-set>, <field-set>, <timestep>. The individual fields are tags, which is why this is called a tagset data model. Figure 18 shows the general structure of this data model. Furthermore, the layout is very similar to a table. The timestamp in the time format forms the most important unit and can optionally represent second, millisecond, microsecond, or nanosecond precision. The tags usually have different data types assigned depending on the column’s content. Besides, there is no limit to the number of tags and fields compared to other TSDBs. Due to the unique InfluxDB data model, data can be queried and processed much faster than with other TSDBs [50].

| Time | Tag 1 | Tag 2 | | Field 1 | Field 2 | |
|------|--------|--------|-------|--------------------------------|--------------------------------|-------|
| | String | String | • • • | String Float Int Bool | String Float Int Bool | • • • |

Figure 18. Tagset data model of Influx [47]

The data stored in the buckets can be processed by so-called tasks at cyclical intervals with the query language Flux and stored again in buckets for further use. There is also

the option of integrating monitoring and alerting applications and visualizing data in dashboards. The visualization of the data in Influx is very limited, so external solutions such as the integration of Grafana are mostly used. These functionalities can be conveniently configured, programmed, and monitored using a web-based interface from Influx [49]. Chapter 3 further clarifies and intensifies the functionalities described above through the use in the context of this master's thesis.

Flux query language

Flux is a data scripting and query language. It aims at querying and manipulating time series data. In the previous versions of InfluxDB, InfluxQL, a query language based on SQL, was used. However, as the functionality increased, InfluxQL reached its limit, so the new language Flux was developed. Above all, it was ensured that this language is useable, readable, composable, testable, contributable, and shareable. Useable means that Flux should be easy to use and highly productive. It should also be readable since programmers usually read more code than they write themselves, representing a considerable time factor. Furthermore, it should be possible to assemble your own functions and libraries. Simple testing of queries is also a focus of development. Finally, Flux should ensure that everyone can contribute and easily share content. This is particularly evident from the open-source architecture; other developers' templates can be adopted if required [49].

“I don’t want to live in a world where the best language humans could think of for working with data was invented in the 70’s”

(Paul Dix, Founder and CTO of InfluxData)

With this quote, Paul Dix, the founder of InfluxData, responded to the question of why not simply using SQL instead of developing a new language. SQL was designed to work with relational algebra and sets. This, therefore, applies less to the flow-based model of time series data. The continuous stream of data must be processed and transformed, which is what Flux is ideal for. Flux is based on JavaScript and has a very different conceptual model than SQL. It represents the result of an open-source community in the area of working with time-series data. Besides, Flux is also characterized by the support of multiple data sources and the ability to cross-compile. Flux can work with another syntax like PromQL, InfluxQL and thus expand the range of its applications. At the same time, it enables an upgrade of existing InfluxDBs without changing the whole system [49, 48]. Finally, a short example will show the advantages of Flux:

```
from(db:”metrics”)
|> range(start:-1h)
|> filter(fn: (r) => r._measurement == “example”)
|> exponentialMovingAverage(size:-10s)
```

This short excerpt shows the formation of an exponential moving average over the data from a bucket with the name “metrics” with the property “example” in a defined time range. The pipe-forward operator `|>` always signals the direction of the data flow. In this example, the data is passed on from the source “telegraf” to the three functions [49].

2.6.2 TimescaleDB and SQL

TimescaleDB is an open-source, non-schemaless time-series database management system designed to extend a relational database. The relational database PostgreSQL serves as the basis, which is a widespread solution in the relational database segment. However, to be able to save more significant amounts of data in a short time with this type of database, an extension by TimescaleDB was published in 2017. This means that data can be read and saved up to 100 times faster than PostgreSQL. The main differences to the underlying database system are: *“Much higher data ingest rates, query performance ranging from equivalent to orders of magnitude greater and time-oriented features”* [47]. In addition to Linux and OS X, Timescale can also work on the Windows operating system. Like influx, Timescale supports several common programming languages, such as Java, Phyton, C, C++, and many more. Timescale is like Influx mainly used in the following areas: Application Performance Monitoring, Industrial IoT, DevOps monitory, and real-time analytics. As already mentioned, SQL is used as the query language [45, 51].

Data model

TimescaleDB works with the relational data model, which was already dealt with in Chapter 2.5.1. As shown in Figure 19, a timestamp is assigned to each new time-series measurement and saved in a separate line. In addition to a timestamp, any number of fields can be added. It can include *“floats, ints, strings, booleans, arrays, JSON⁴ blobs, geospatial dimensions, date/time/timestamps, currencies, binary data, or even more complex data types.”* [52]. Indices can be created for all of these fields. Besides, several fields can be combined into one index. The choice of indices depends mostly on the subsequent use of the data. In addition, secondary tables can be assigned to individual fields via a foreign key, in which additional metadata⁵ can be stored [52].

The advantages of this system are flexibility concerning the width of the table. This depends on how much data has to be saved. Furthermore, queries can be significantly

⁴ “JSON (JavaScript Object Notation) is a lean data exchange format that is easy to read and write for humans and easy to parse (analyze data structures) and generate for machines.” [78]

⁵ “Metadata, at its most basic level, is ‘data about data. Metadata should be structured, organized information about an object such as its source, scope, physical or digital characteristics, context, or any other details about the object itself.’” [76]

accelerated by freely choosing indices. The outsourcing of non-standardized metadata to a separate table also improves the system's performance. The rigid schema of the database also has a positive effect here. In addition to some advantages, the strict scheme also offers a decisive disadvantage: at the beginning of a new application, a scheme must be selected, which is very difficult to change later [52].

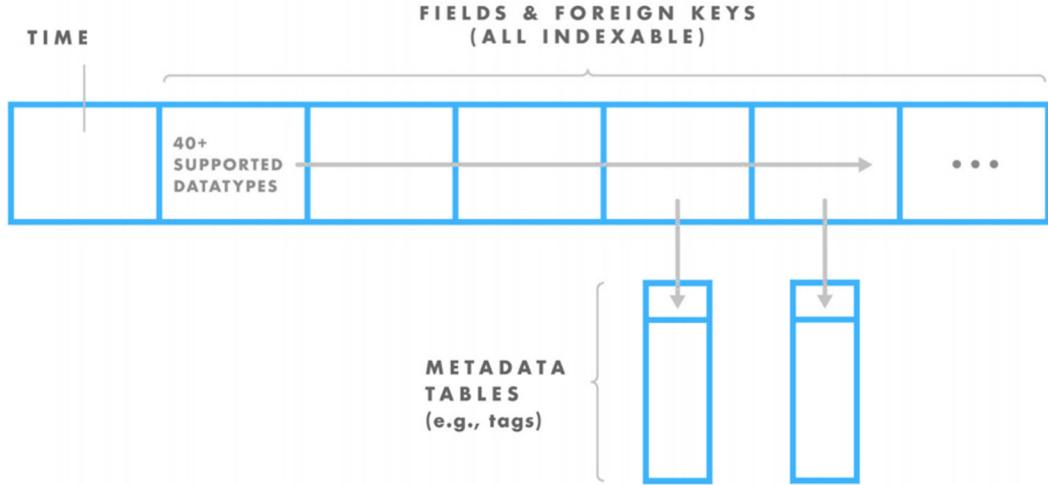


Figure 19. Relational data model for time-series data [47]

Architecture

To store a large amount of data despite the relational database system, Timescale intervenes deeply in the PostgreSQL engine. In relational databases, data is stored according to the scale-up principle, i.e., new data is stored on exactly one node, the data table. This node has to save all new data and can therefore quickly reach its capacity limits in the event of a high volume of data. Timescale uses hypertables to store the data, representing the abstraction of many individual tables, so-called chunks, which can be seen in Figure 20. These chunks correspond to a specific time interval and partition keys. This is called automatic space-time partitioning and enables horizontal scaling [47].

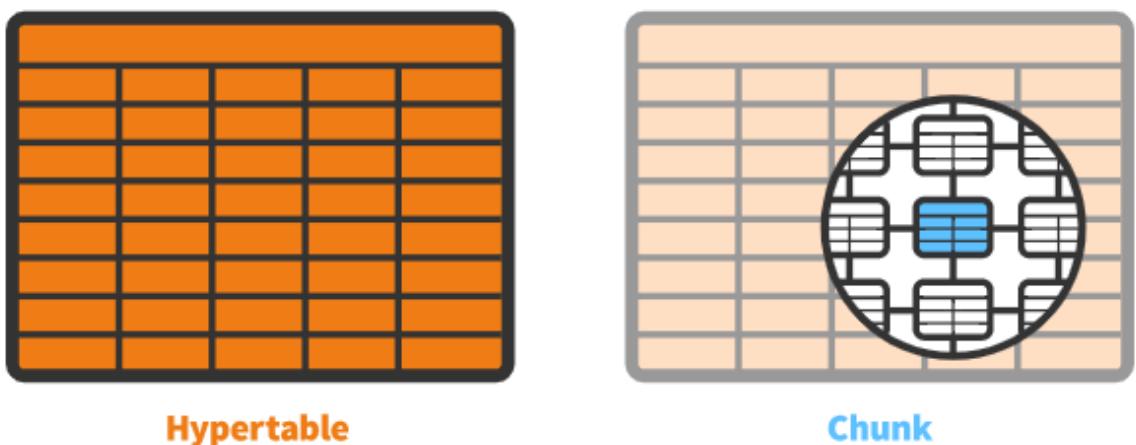


Figure 20. Schematic representation of a hypertable and the belonging chunks [47]

The stored data is partitioned into chunks by several dimensions, one of them represents time. According to the scale-out principle, these are distributed over a cluster to many individual nodes. These individual nodes, in turn, can store up to 1000 chunks according to the scale-up principle. Due to a large number of nodes, a significantly larger amount of data can be processed simultaneously. Because of the abstraction of these nodes and the stored chunks, the hyper table looks like a standard data table to users [47].

Structured query language

SQL, Structured Query Language, is the essential standard language for database programming. In 1987, SQL was standardized for the first time by ANSI (American National Standards Institute) and ISO (International Organization for Standardization), which was adapted continuously to technical change. Today the SQL3 standard regulates all details of the query language. The database providers only make minor changes to adapt them to their systems [28, 53].

Figure 21 shows a table on the left based on the relation model, in which information on fictitious employees is stored. Due to the number of contained tuples, the query or manipulation of the data can be done in a quantity-oriented manner. The result of a search is always a table. SQL is a descriptive language, i.e., the desired result is described, whereas procedural database languages require the necessary calculation steps. This architecture means that the search and access methods required for the query are defined by the system and not by the user. This significantly increases the usability of the query language [7].

| EMPLOYEE | | |
|----------|---------|-----------|
| E# | Name | City |
| E19 | Stewart | Stow |
| E4 | Bell | Kent |
| E1 | Murphy | Kent |
| E7 | Howard | Cleveland |

| Name |
|--------|
| Bell |
| Murphy |

Results table

Figure 21. Example table and results table (based on [7])

A short query should be presented as an example of the descriptive approach. In full form, this describes the desired result: “Select the names of the employees living in Kent.” This query results in the table shown in Figure 21 on the right. The implementation of the query with SQL would be [28]:

```
SELECT Name
FROM EMPLOYEE
WHERE City = “Kent”
```

2.7 Data visualization

Decisions at all management levels are mostly based on available information. While there was often insufficient data available in the past which led to a potentially incorrect decision, the opposite is often the case today. The number and scope of data sources are continually increasing. Therefore, more data is available today than ever before. Suitable information visualization can counteract this multitude of data and support the decision-maker in their work. *“With the help of dashboards [...] in particular, the most important and decision-relevant information for the user can be visually displayed and aggregated at a glance using various diagrams and tables.”* [54]

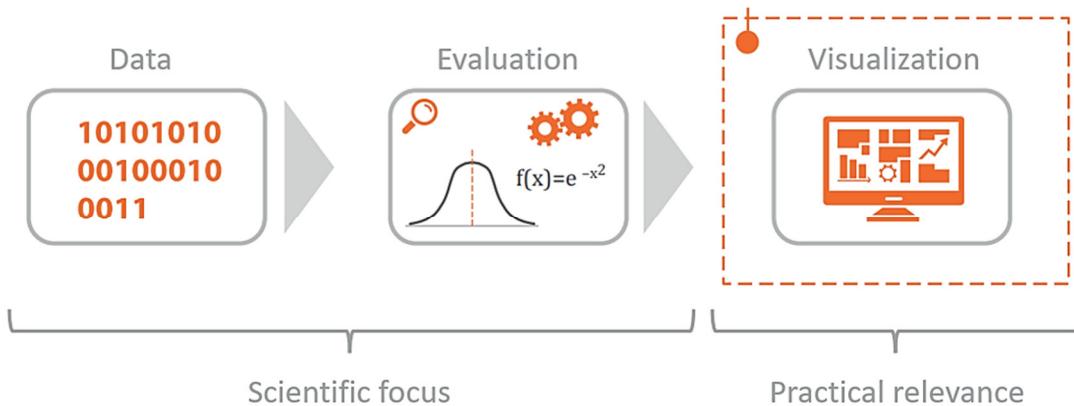


Figure 22. Visualization is an essential step in data analysis (based on [54])

Data analysis consists of three steps, as shown in Figure 22, the generation of data and the subsequent processing of the data. At this point, information arises that should be effectively visualized, especially concerning practice work. The visualization of the data makes it visible and understandable and can lead to new insights for the viewer. A successful visualization is, therefore, primarily characterized by effectiveness and efficiency. Effectiveness in terms of understanding the information and effectiveness in terms of information absorption speed. Figure 23 shows a simple example: all numerical values should be counted equal to “5”. In the left visualization, this presents itself as very difficult. On the other hand, in the right visualization, counting can be made much more quickly by simply using color [54].

| No Attribute | Attribute Color |
|--|--|
| 9 2 6 5 4 8 6 2 3 4 2 2 9 6 5 4 1 8 7 1 9 8 3 2 5 4 3 2 2 5 6 1 4 5 9 3 3 4 2 1 4 5 7 4 5 4 6 2 9 1 6 2 | 9 2 6 5 4 8 6 2 3 4 2 2 9 6 5 4 1 8 7 1 9 8 3 2 5 4 3 2 2 5 6 1 4 5 9 3 3 4 2 1 4 5 7 4 5 4 6 2 9 1 6 2 |

Figure 23. Illustration of perception using the attribute color (based on [54])

O'Donnell and Zimmer have defined eleven rules as principles for creating visualizations based on perception psychology. It should be noted that the nature of the data is mostly responsible for the form of the visualization [55].

- Rule 1: Less is often more → Better understanding of data instead of eye-catchers
- Rule 2: Good relationship between data and ink (Ed Tufte)
- Rule 3: The type of graphic should match the data and the statement
- Rule 4: Design axes consciously and be aware that you are designing them
- Rule 5: Identify the central message of the graphic and let it guide you
- Rule 6: Choose colors wisely
- Rule 7: Consider the limits of human perception
- Rule 8: Consider the target group and do not assume too much knowledge
- Rule 9: Make your graphics reader-friendly
- Rule 10: Limit the number of categories in graphics if possible
- Rule 11: Use common formats and pay attention to reusability and traceability [55]

These rules serve as the basis for implementing an effective and efficient visualization, as it should also be implemented in the context of this master's thesis. The open-source program Grafana is used as a software tool for practical implementation, briefly presented below.

Grafana is a cross-platform and open-source application in data visualization, which can process data from various data sources such as InfluxDB, TimescaleDB, MySQL, PostgreSQL, Prometheus, and Graphite. Grafana is based on the Go programming language and supports using multiple query languages to process data, including Flux and SQL. This is guaranteed by a large number of available plugins. The essential functions and properties of Grafana consist of visualizing, alert, unify data, open-source, extend and collaborate. Plugins can be used to extract data from up to 30 different sources and display them in dashboards. The time range to be viewed can be set on the dashboards. Furthermore, the dashboards can be filled with so-called panels. These panels display the desired data in the preferred form. As Figure 24 shows, it can be chosen from many different visualizations and Flux or predefined functions can be used to bring them into the desired shape [56].

Practical examples are not shown at this point. These are carried out and described in the following chapters based on implementing a time study.

Literature review

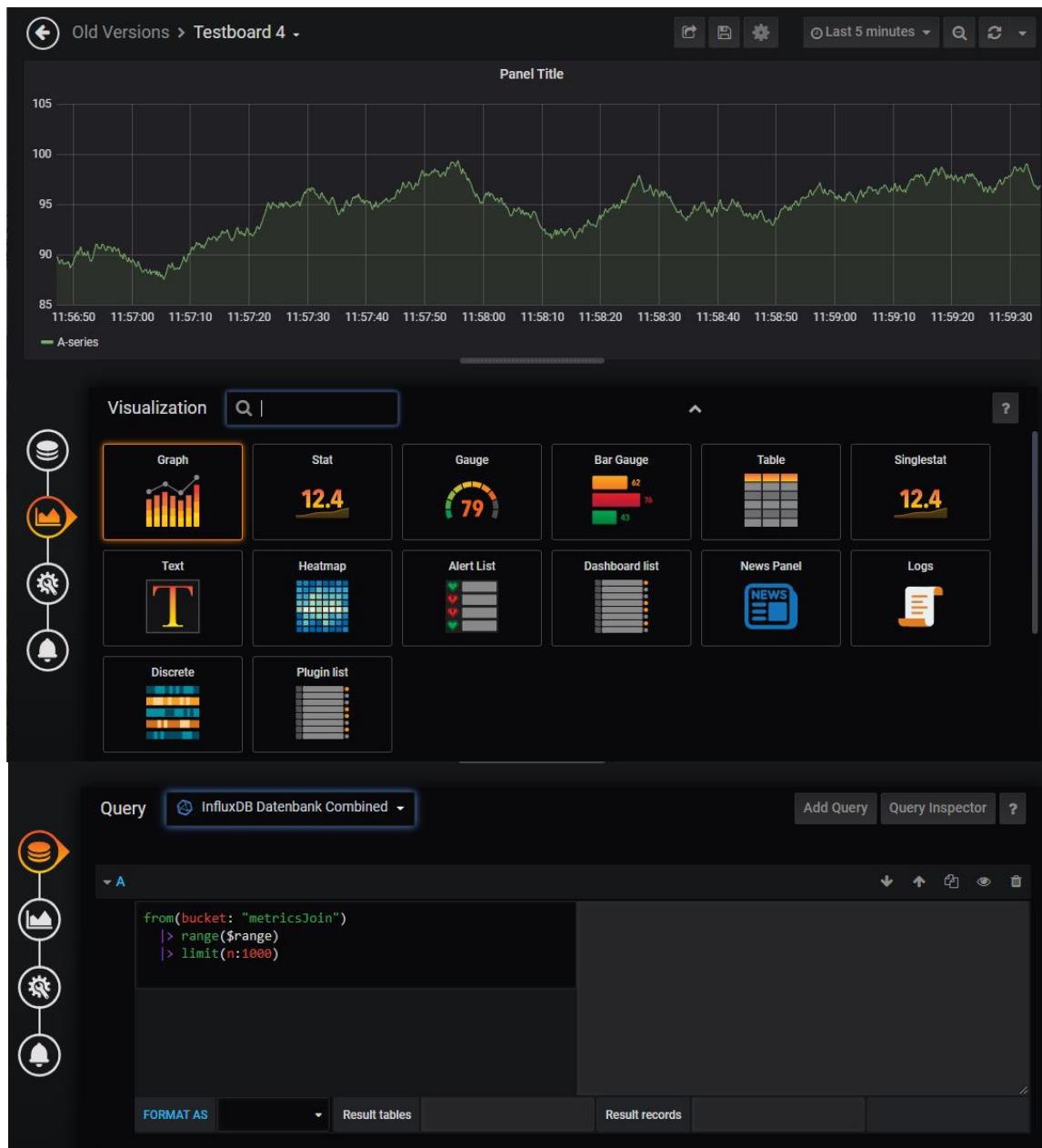


Figure 24. Exemplary representation of the possible visualizations of data in a dashboard panel

2.8 Data analytics

The term “data analytics” has been used more and more since the early 2000s. The encyclopedia techopedia describes the term “data analytics” as follows: *“Data analytics refers to qualitative and quantitative techniques and processes used to enhance productivity and business gain. Data is extracted and categorized to identify and analyze behavioral data and patterns, and techniques vary according to organizational requirements.”* [57] The computer science professor, Dr. Thomas Runkler at the Technical University of Munich, provides another apt definition: *“Data analytics is defined as the application of computer systems to the analysis of large data sets for the support of decisions.”* [58] Data analytics can generally be understood as an interdisciplinary field. Methods from various other disciplines are often used, such as operations research, machine learning, AI, the recognition of patterns, and statistical methods. As shown in Figure 25, data analytics is divided into four main phases: preparation, preprocessing, analysis and postprocessing. Besides, the graphic shows the main methods belonging to the four phases [58]. Some of them will be used during this work.

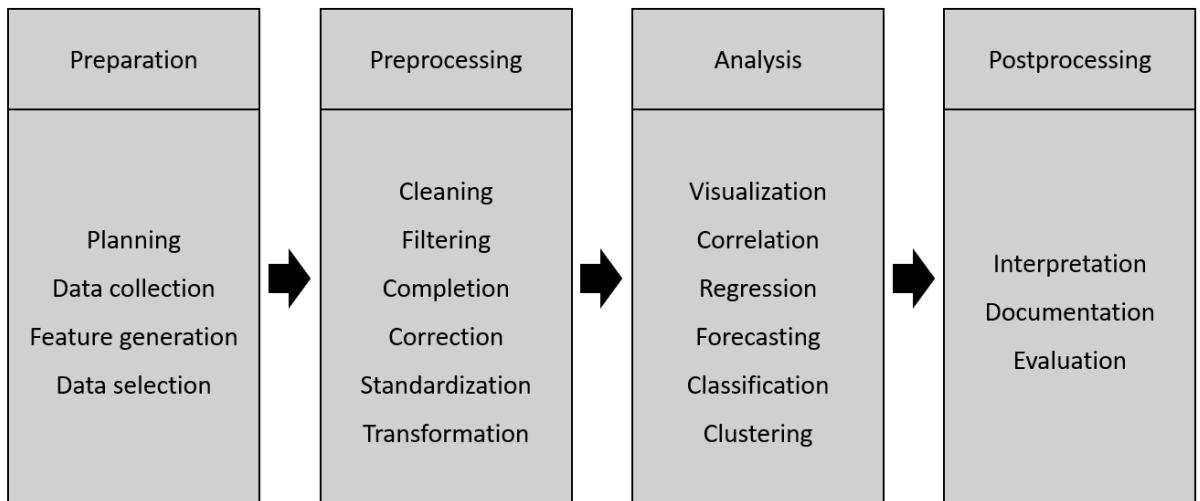


Figure 25. Phases of data analysis projects (based on [58])

Most of the time, the term data analytics is used for the preprocessing and analysis phases [58]. In this chapter, the essential basics of these phases are presented. The time series decomposition, the so-called component model, is discussed, and various mathematical methods are assigned to these components. Selected methods are then examined in more detail.

2.7.1 Component model

Time series can usually be broken down into several components. This is described in the component model for time series decomposition. A time series consists of a smooth component g_i , a seasonal component s_i and an irregular component ε_i , which are linked by a function H [59].

$$y_i = H(g_i, s_i, \varepsilon_i), \quad i \in \{1, \dots, n\} \quad (2.1)$$

Smooth component g_i : Describes the long-term basic direction in which a time series develops and thus describes the curve trend. As a rule, trends change only very slowly, so the curve is comparatively smooth [60].

Seasonal component s_i : These are regular, cyclical fluctuations around the trend of a curve. These are described by their duration and deviation from the trend. Furthermore, the seasonal components' duration can range from very short intervals such as hours to years [60].

Irregular component ε_i : The third component of a time series is called the irregular component. This can be both the occurrence of one-off variables that change the time series and irregular but repeated variables. This category also includes data from position sensors, for example [60].

The function of the three components described in equation 2.1 is usually linked additively or multiplicatively. With an additive link, the components influence the time series independently. With a multiplicative link, they act not independently on the time series. The time series described in this work has an additive link shown in equation 2.2 [59]. This fact will be explained closer in Chapter 3.

$$y_i = g_i + s_i + \varepsilon_i, \quad i \in \{1, \dots, n\} \quad (2.2)$$

A large number of descriptive and model-free methods are available for processing time-series data. The selection of them depends on the desired result of the processing and the considered component. A deterministic process is assumed here, i.e., each value of the time series is causally dependent and determined on other, previous values. The main methods are from the areas of transformation, aggregation, regression, filtering, and moving averages. Some methods are based on a stochastic process, especially in forecasting procedures called the Box-Jenkins method. The most common methods are autocovariance and autocorrelation functions, analysis in the frequency domain, and ARMA (autoregressive moving average) methods with their further development ARIMA (autoregressive integrated moving average) and ARMAX (autoregressive moving average with exogenous inputs) [60].

In this master's thesis, more than one component is examined. The irregular components are considered in the IoT sensor data's first analysis. However, the analysis of working time data over a more extended period can be assigned to the analysis of a smooth component. It is also examined whether a seasonal component is present. Therefore, the data analysis represents a combination of time series analysis methods for events, trends, and forecasts.

2.7.2 Selected methods

In the following, data processing methods used in this thesis are presented as examples. Since measurement data often contain undesirable signals, such as noise or isolated outliers, filters can be a simple solution. These include reduction filters and difference filters. A reduction filter is primarily used to reduce data by algorithmically accumulating points a_i, a_{i+1}, \dots, a_j until the distance between a_i and a_j exceeds a clearly defined limit. The interval's mean value replaces all points that are in this interval according to equation 2.3. The next interval then starts with a_{j+1} . A reduction filter simultaneously reduces the noise in sensor signals [61].

$$\frac{1}{j-i+1} \sum_{k=i}^j a_k \quad (2.3)$$

Difference filters represent a possibility for trend elimination. This enables the option of obtaining an original series. The original series can be extracted through the constant formation of differences between neighboring time series values. The order of the difference can be selected as required. Equation 2.4 shows a first-order difference filter [62].

$$\Delta x_t = x_t - x_{t-1} \quad (2.4)$$

Another important method is the aggregation of data. The time series is divided into defined time windows. For these time windows, the average, median, sum, minimum or maximum values can be calculated. This means that effects similar to those of a reduction filter can be achieved [62].

One of the most important methods for determining trends is the single moving average. It is defined as follows: "*With the moving averages method, the value of the smooth component at a specific point in time is approximated around this point in time by the arithmetic mean of the observed values in a time window.*" [59] A distinction must be made between odd ($2k+1$) and even ($2k$) orders for calculating the single moving average. Equation 2.5 describes an odd order, i.e., an odd number of time series points is considered. Equation 2.6, on the other hand, considers a single moving average of even order

so that only half of the first (y_i) and the last (y_j) time series point is included in the calculation. The order's selection increases the smoothing. The higher the order, the higher the curve's smoothing [59].

$$y_i^* = \frac{1}{2k+1} \sum_{j=-k}^k y_{i+j}, \quad i \in \{k+1, \dots, n-k\} \quad (2.5)$$

$$y_i^* = \frac{1}{2k} \left[\frac{1}{2} y_{i-k} + \sum_{j=-k}^k y_{i+j} + \frac{1}{2} y_{i+k} \right], \quad i \in \{k+1, \dots, n-k\} \quad (2.6)$$

In addition, there are several moving average methods based on this, including exponential moving average (EMA), double EMA, triple EMA and moving exponential average (MEA). Exponential moving averages are not moving averages in the mathematical sense. Rather they are exponential averages (EA) with a specific weight according to equation 2.7, which is also called exponential smoothing. The weight is determined by selecting the considered time frame (N-day), shown in equation 2.8, and significantly influences the EMA result.

$$y_n^* = (1 - \alpha)^{n-1} x_1 + \alpha \sum_{k=0}^{n-2} (1 - \alpha)^k x_{n-k} \quad (2.7)$$

$$\alpha = \frac{2}{N+1} \quad (2.8)$$

$$y_n^N = (1 - \alpha)^{N-1} x_{n-N+1} + \alpha \sum_{k=0}^{N-2} (1 - \alpha)^k x_{n-k} \quad (2.9)$$

Double EMA and triple EMA represent variants of the EMA with a more substantial weighting of the exponential component. In comparison, the moving exponential average (MEA) represents the real moving average since the time window is directly included in the calculation according to equation 2.9. All these MA variations differ essentially in the past data's different weighting. It is up to the user to choose them sensibly.

Another method for determining trends is the further development of exponential smoothing in the holt winters method. Similar to MA variants, the weighting of past values is varied. As a further development, exponential smoothing is combined with season and growth [63].

2.9 Time structure analysis

Time structure analysis methods are time management measures. They provide time data on the start and end as well as the duration of operational activities. This time data serves as a basis for various purposes, including planning, management, and control in a production environment, remuneration and wage differentiation of employees, determination of requirements and job evaluation, job design, and the necessary work instructions [64]. These examples represent only a part of the possible uses of time data. The potential benefits can be management-related, employee-related, product-related, or production-related. *“Time management is understood to mean the management of all the times required in the company for employees, work / operating equipment, and work objects. The tasks of time management range from the determination of time data for individual work processes to deadline and schedule planning to scheduling and scheduling. The latter compares the planned target times with the actual times that arise and, if necessary, intervenes in the production process or the scheduling.”* [5]

The foundations for time studies were laid by Frederick Winslow Taylor (1856-1915). For his time studies, he carried out time measurements for previously defined work steps, which he used as the basis for setting target working times. Since the results were subject to a wide range due to employees' different capabilities, he also smoothed the results. Today this methodology is no longer used because it does not consider aspects such as the monotony of work execution or one-sided burdens. [65] Several methods are known today for determining time data, which can be classified into continuous observation, sample observation, and computational-analytical procedures. As part of continuous observations, manual time measurement of activities is carried out by the worker himself or an external employee. The multi-moment method is used in the field of sample observations. Systems with predetermined times are considered computational-analytical processes. These methods only provide target times, whereas continuous observations and sample observations record actual times. According to a survey by the Institute for Applied Work Science e.V., time recording corresponding to REFA from the field of continuous observations is the most widely used method for determining time data. Around 50% of the surveyed companies stated that they primarily use the REFA methodology to carry out time studies [5]. REFA is an association for work studies in work design, company organization and company development. REFA develops methodologies for these areas, offers training and consulting support [66]. REFA describes its core purpose as follows: *“The work of the association serves to promote, develop and maintain a competitive economy, administration and service. The promotion and further development of humane work for those employed in these areas are of equal importance.”* [67] Due to the high relevance of the REFA time recording method in today's industrial environment, the REFA time recording method's basics are presented below.

Literature review

The general goal of a REFA time study is to determine target times for work processes. The evaluation of actual times serves as the basis for this. A prerequisite is the reproducibility of the data, which means that the work process must be writable, and the working conditions must be known. Furthermore, the actual times must be technically measurable. To receive statistically relevant data, the work processes must be cyclical, i.e., repeatable activities [64]. Figure 26 shows the schematic sequence of a time recording according to REFA, which is divided into eight steps. The first step should be defining the intended use of collected time data and checking the previously mentioned requirements. In step 2, preparatory activities are carried out. These tasks include, for example, the notification of affected employees. In the next step, it must be chosen between progress and individual time measurement. With the progress measurement, the continuous-time values are noted, with the individual time measurement, however, only the duration of the individual work steps. Step 4 involves choosing the timing device. The time measurement can be done with a stopwatch or an electronic alternative. In this work, the sensors and the data processing unit of the factorycube are used for this purpose. In the next step (5), a timesheet must be selected regarding the data collection scope. Then in step 6, the tracked activity must be described in more detail and divided into process sections. These process sections serve simultaneously as the start and endpoint of the time measurements. The accurate description of the activity enables the measurements' reproducibility and creates the necessary transparency, especially regarding management-oriented remuneration. In step 7, the sequence sections' actual times are measured and evaluated in the subsequent step 8 [5, 68]. The evaluation of the data takes place according to the following procedure: *“Check time recording for correctness and completeness, calculate actual individual times, statistical evaluation, calculate target times (possibly normal times), calculate the default time, document possible approaches for improvement measures, record the assessment of the stress and strain situation.”* [68]

Literature review

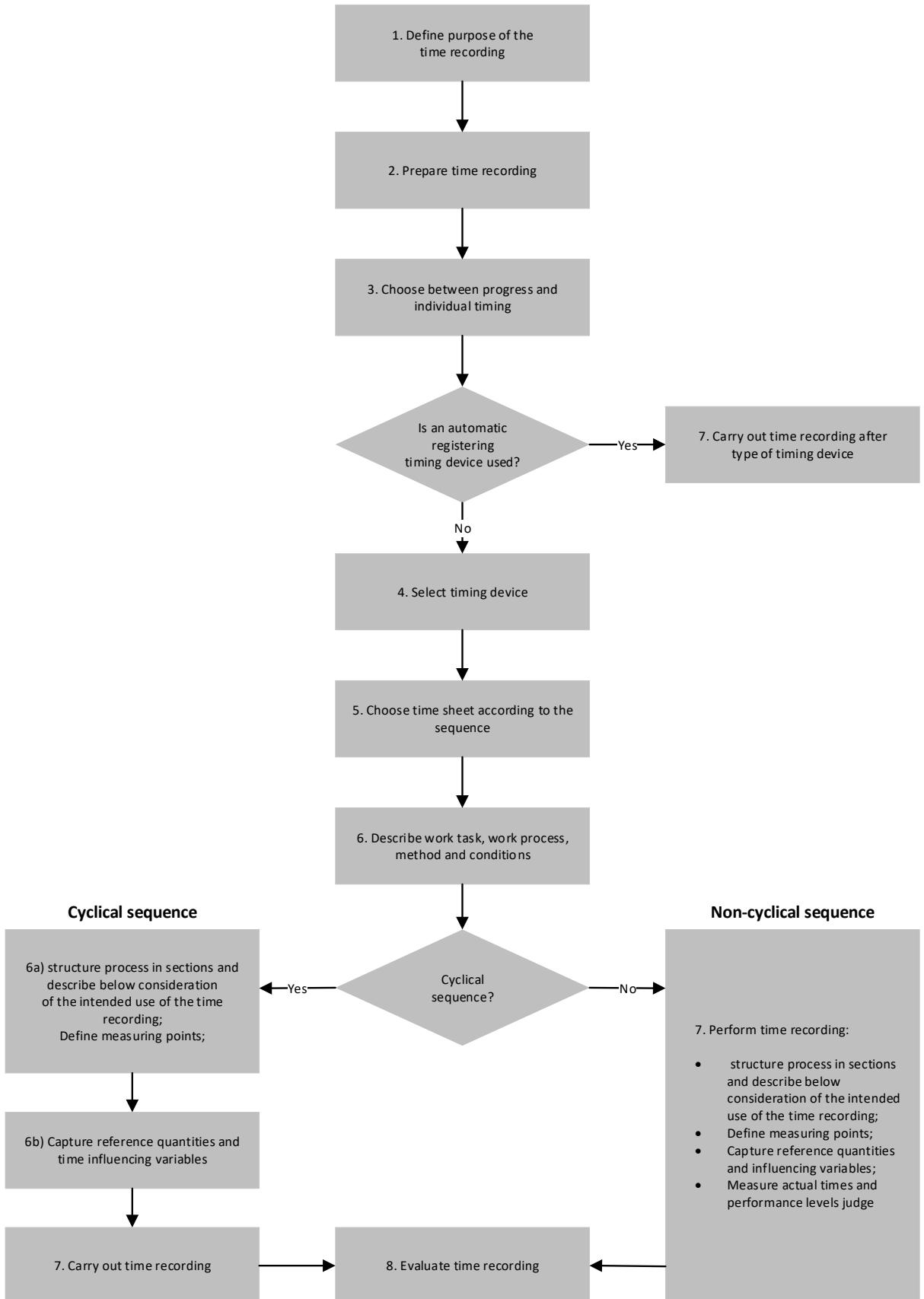


Figure 26. REFA standard time recording method (based on [5])

2.10 Summary

In this chapter, a literature review necessary for understanding this master's thesis was done. The basics of training development in learning factories and an overview of the topics Internet of Things and Big data were conveyed. Furthermore, the RFID technology used in the LEAD factory was presented, and its functions explained. An essential point in this chapter is the introduction to databases. The general term "database" was defined, but the different database types, such as relational and NoSQL database, were examined. Time series databases and the time series data format were also presented as special database types. Furthermore, the time-series databases of influxdata and timescale were described in this context. Additionally to these databases, their query languages Flux and SQL were presented. The focus was also placed on the basics of data visualization and the Grafana visualization tool. To complete the basics in the area of time series data, fundamental data analytics methods were presented specifically for time series data. Finally, there was an introduction to time structure analysis according to REFA.

In summary, it can be conducted that there is a broad scientific basis for the covered topics. Nevertheless, the literature review clarifies that research gaps can be found in some areas. On the one hand, this concerns the practical data work with the different software stacks of the factorycube, which will be explained in more detail in the following chapter, as well as the data processing of event peaks in time series data. Additionally, due to the query language Flux's novelty, there is less information about the implementation of specific features. Besides, implementing a time study with IoT devices and time-series data is an exciting alternative due to the large and highly precise amount of data and the associated possibility of long-term analysis.

3 Methods

In this chapter, a digital time series-based solution for performing a time recording study is to be developed. A setup consisting of IoT sensors, the IO-Link system, and a factorycube with two different software stacks from the tech startup Industrial Analytics is used for implementation. For this purpose, the evaluated literature research results in the previous chapter must be linked and completed at the mentioned points. An overview of these components required setup and interaction, especially the differences between the used software stacks, should be presented initially.

The implementation of the digital time study is based on a simplified waterfall model for software development. This model consists of five phases, which are run through like a waterfall. The results always fall into the next phase. The phases are analysis, design, implementation, testing, and operation [69]. The first phase addresses the definition of software requirements in the form of current and target state. The phase can be assigned to the experimental setup description, the preparation of the time recording, and software requirements described in this chapter. The subsequent design phase deals with system design and specification in the form of the software architecture, which is represented in a BPMN. The two different software stacks with their different data models have to be taken into account, and two different architectures have to be designed. The phases of implementation, testing, and operation of the two software variants are presented in Chapter 4.

3.1 Experimental setup

According to Figure 27, the used system consists of the factorycube (1), an IO-Link master (2), and various IoT sensors (3). The factorycube represents a black box in which a time-series database system and the Grafana visualization program are hosted. A more detailed explanation of this follows in the next chapter. The data transfer from the sensors via the master to the factorycube occurs via IO-Link. *“An IO-Link is an automated communication system in which sensors and actuators are integrated. Actuators are drive technology components that can convert electrical signals into mechanical movements. The system works with standardized communication (IEC 61131-9). The individual components of the system are called IO-Link masters (control unit) and IO devices (sensors, actuators).”* [70]

The existing system components are classified in the IoT Reference Model from CISCO to define a systematic structure. In Level 1, the Physical Devices and Controllers level, the system’s IoT sensors, and the RFID sensors and energy measurement system in the LEAD factory are located as “things”.

Methods

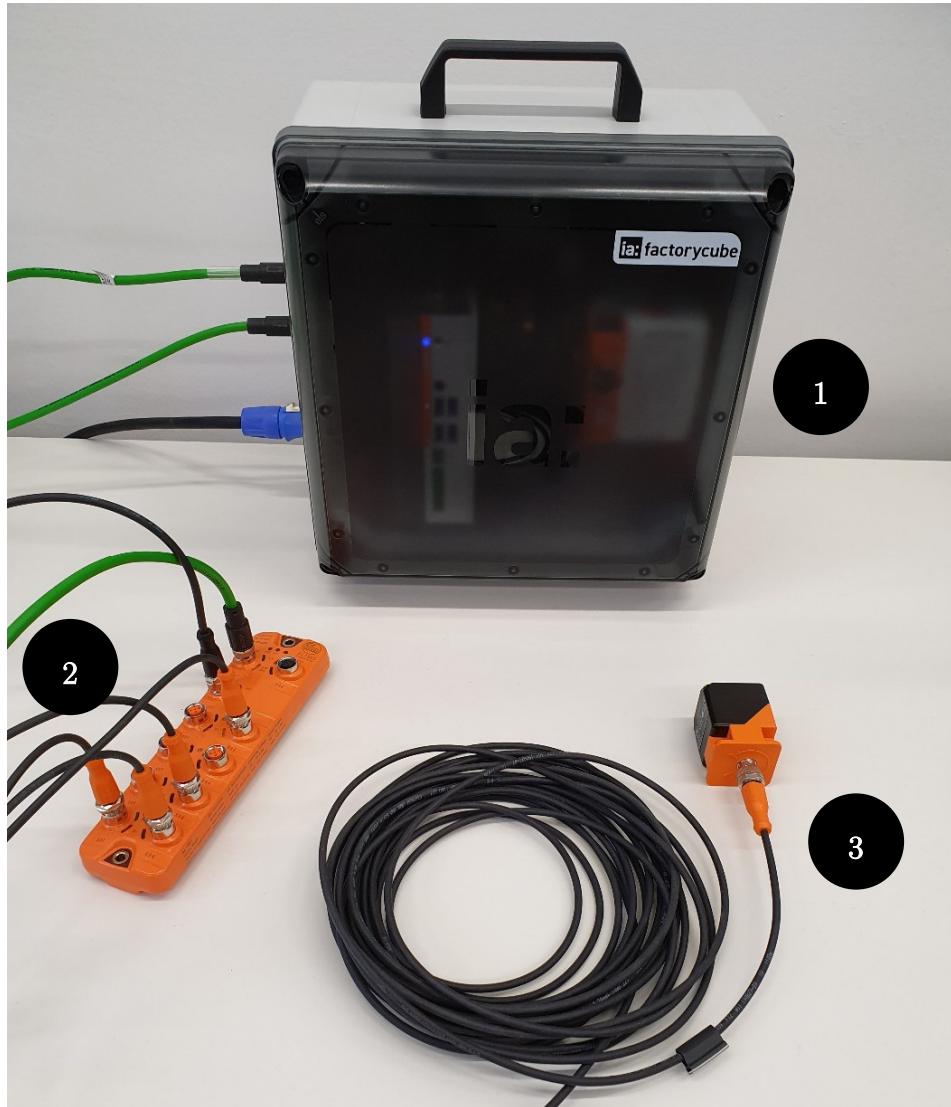


Figure 27. Basic setup of the factorycube

The here determined data is transmitted using the connectivity level. In the case of IoT sensors, this is done by the IO-Link communication system; with RFID sensors, the data is transmitted via JSON. The Edge Computing level is optional and does not exist in this setup. Data accumulation and the necessary storage of the data according to level 4 are accomplished in the case of the IoT sensors by the InfluxDB or TimescaleDB and for the RFID data by the relational DBMS Microsoft SQL server. The subsequent data abstraction according to level 5 is done by combining all existing data in InfluxDB or TimescaleDB and processing them via so-called tasks in influxdata's DBMS or via Node-red with Timescale. The aggregated data is also stored in this database. In level 6, "Application", the processed data is visualized via reporting, analytics, and control functions. This visualization takes place using Grafana. Before this, only the data from the Influx or Timescale database has to be transferred to Grafana. The final 7th level, "Collaboration & Processes," is no longer part of this work.

3.1.1 Factorycube

With the factorycube, as shown in Figure 27 (1), the Aachen startup ia: industrial analytics has been offering a portable, modular plug-and-play system since 2019, with which data from a wide variety of sources can be collected, processed, and visualized. The factorycube provides the hardware and software to evaluate data for applications in the areas of performance management, quality control, maintenance, and condition monitoring. This system is mainly used for digitizing analog production machines [71]. The factorycube can be understood as a “black box” since the programming is usually carried out by industrial analytics. Due to the new form of application, which is carried out in this master’s thesis in the form of a digital time recording study and the novelty of the product and associated problems, regular exchange with the development team of Industrial Analytics took place. Continuous feedback enables constant improvements to be made to the system. In the course of this master’s thesis, a switch was made to a new software stack due to repeated errors. The two software stacks used are described below.

Due to Industrial Analytics’ closed business model, it was not possible to obtain technical documents or data for the first software stack. That changed later with the introduction of the second software stack. The familiarization with the internal structure of the factorycube had to be carried out through step-by-step testing. The factorycube essentially consists of a 24 V DC power supply module, a RevPi Core 3, which serves as a powerful industrial computer, a hard drive, and a router. A hub for distributing the data streams is also built-in. The 24V DC power supply module supplies all essential components with their operating voltage. The system can be accessed wirelessly via the router. At the same time, data from other sources can also be transferred. The resulting sensor data is sent via the IO-Link master to the IO-Link gateway of the RevPi. In combination with the hard disk, this industrial computer forms the system’s center, as the DBMS runs on it and the data is saved. The used RevPi uses Ubuntu as the operating system. Ubuntu is based on Debian GNU / Linux and is, therefore, a Linux distribution. This means that the RevPi has a very easy-to-use operating system, which can be accessed via an SSH connection and an IP address. The “Docker” software to implement container visualizations works on this operating system. In a regular operating system, programs can access all system resources. Applications can be isolated through the use of Docker. This means that the desired system resources are assigned to containers and thus to the applications they contain. Basically, containering is used for the efficient use of system resources. Like the RevPi itself, the containers can be accessed via SSH connection and IP address. In this container, the applications InfluxDB (stack 1) and Grafana are isolated, which were already discussed in the previous chapter. In software stack 2, the TimescaleDB database and Grafana are located not locally on the factorycube. They are located on the ia: server. Therefore, the processed data must be transferred to the server via a Message Queuing Telemetry Transport (MQTT) broker by using an internet connection. Besides, there are other differences between the stacks

Methods

concerning the practical implementation of the various tools, especially the introduction of k3os (kubernetes), which should only be explained in this work if necessary.

3.1.2 IO-Link sensors

The following sensors were available to perform a time recording study, no matter which software stack is used. On the one hand, IO-Link sensors are connected to the factorycube via an IO-Link master (Figure 28), as well as sensors that are already available in the LEAD Factory and stored in a relational database. It is essential to use these sensors in the best possible way, according to their functionalities, in the context of the time recording study.

Inductive sensor IM5173 (1): In the sensor, a coil and a capacitor form an LC resonant circuit are installed. If an inductive object penetrates the sensor field, eddy currents arise and remove energy. The sensor registers this and outputs a specific induction value, which depends on the distance and geometry of the object.

Optical distance sensor O5D100 (2): The sensor measures distances of 0.03 to 2 meters using a laser and the associated time-of-flight measurement. The distances in cm are transmitted as output values. The time of flight measurement works for almost all objects and surfaces.

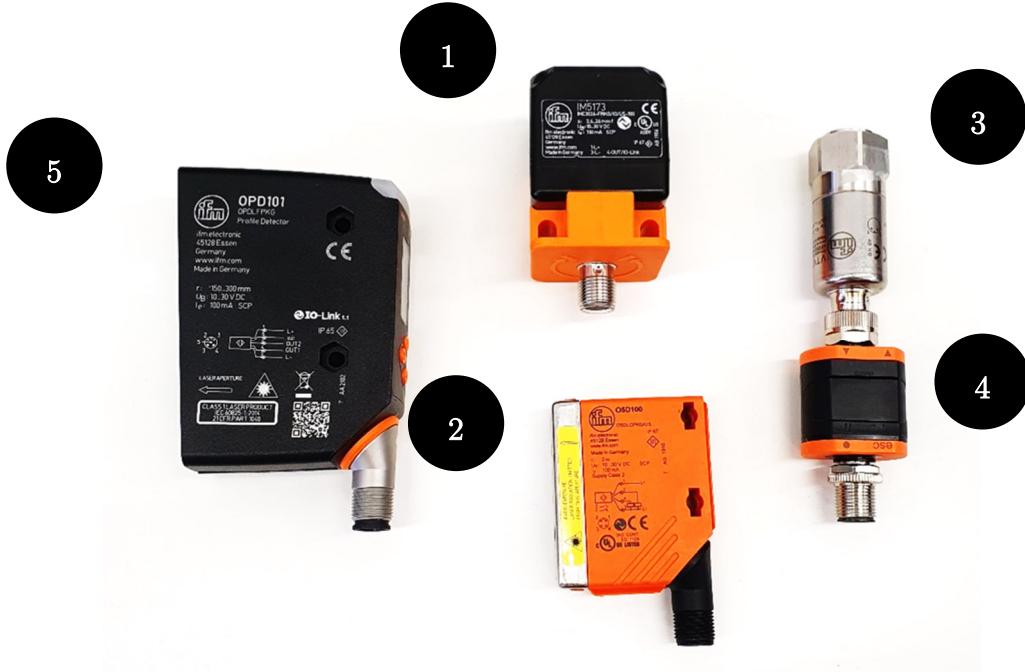


Figure 28. Used IO-Link Sensors

Methods

Vibration transmitter VTV122 (3) with AD converter DP2200 (4): The vibration transmitter monitors a system's vibration based on the measured and evaluated vibration speed. The output takes place via a linearized analog output to the AD converter. This converts the resulting analog measured values into IO-Link.

PMD Profiler OPD101 (5): A height profile is created along a projected laser line. The measurement is carried out in the same way as for the O5D100. The PMD Profiler continuously measures the current height profile and compares this with the saved height profiles. The sensor's output values are in the range of 0 to 100%, which indicates the percentage of the data points valid in the measured height profile compared to the stored profile.

3.1.3 LEAD-Factory sensors

RFID-System

Figure 29 shows the LEAD Factory production's schematic structure in a fully digitized state. The five existing workplaces, WP1 to WP5, are arranged in a U-cell using the line principle. A total of four employees operate the workplaces. WP3 and WP4 are permanently occupied, while workplaces 1, 2, and 5 are so-called jumper workplaces. The scooters' assembly begins at station one and goes through the other workplaces one after the other. In the end, the finished scooter is packed and stowed on a pallet. As described in Chapter 1, this process is optimized concerning lean methods. RFID tags and the associated antennas are used throughout the production process.

Each scooter receives an individual RFID tag at the start of assembly, which is scanned by an antenna at each workstation. This means that every scooter can be tracked in the production process. Furthermore, the material flow is monitored with the help of RFID. There are slide racks for material supply at each workstation. All provided material types are assigned with an RFID tag on their material boxes, which the employee scans as soon as the material box is empty. If a component is defective, this is also recorded by activating an RFID tag. All resulting RFID data are transferred to a database and thus form the basis of a manufacturing execution system (MES). These data should also be used in the context of this master's thesis, as this creates additional measuring points for a time recording study and further refines it.

Energy Meters

The LEAD Factory has an energy monitoring system that uses power plugs to measure the electrical energy used in the learning factory. The power plugs are integrated into all electrical consumers' circuits and send the determined consumption data to the LEAD

Methods

Factory database system. By monitoring the consumed energy, the use of electrical devices and their time and duration can ultimately be determined in further processing within this master's thesis framework. As with RFID data, this information serves as additional measuring points for a time structure analysis.

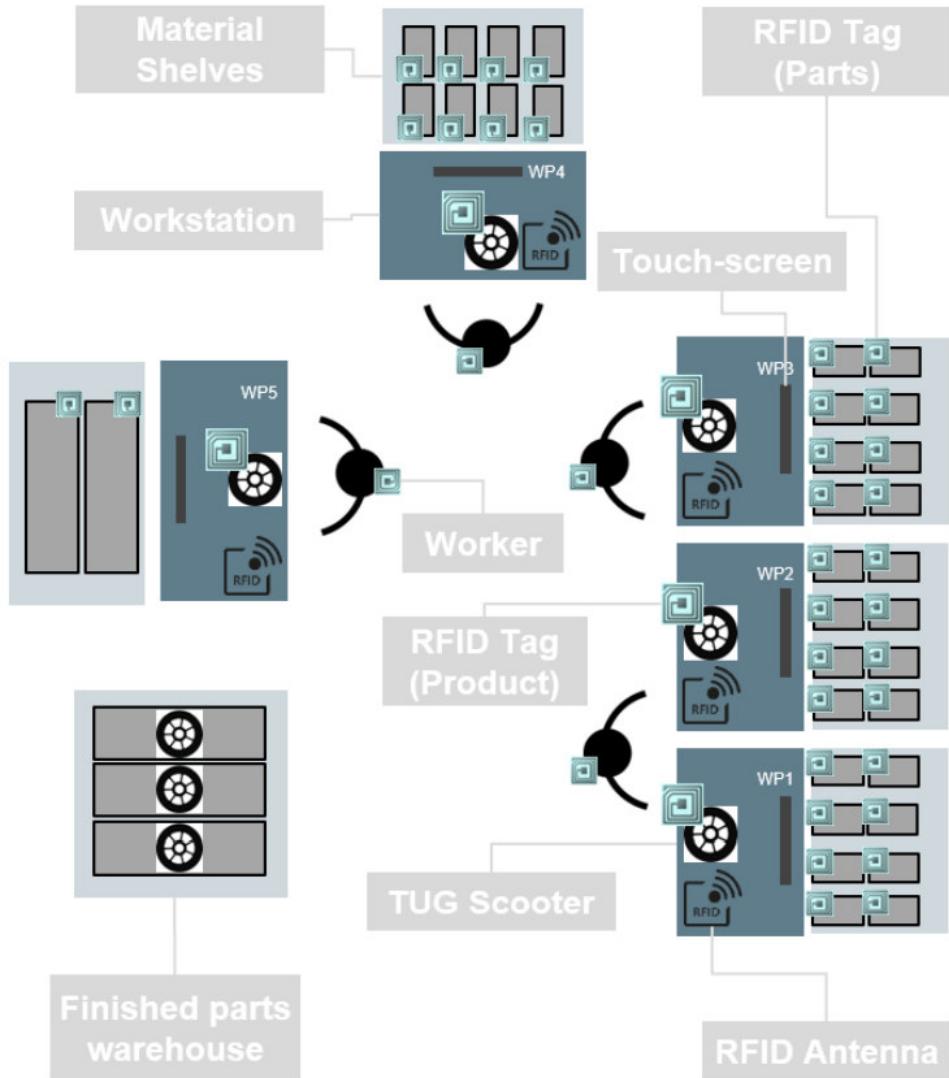


Figure 29. Manufacturing process of the digital state in the LEAD Factory [72]

3.2 Preparation of time recording

To carry out a time recording, preparations must be made following chapter 2.9. For this purpose, the reason for the time recording must be stated at the beginning. The time recording is carried out to identify process defects and the resulting optimization potential. The recording type is individual times, which were chosen compared to progress times due to the better visualization possibilities. The setup, consisting of sensors and factorycube, functions as a timing device, which automatically executes and visualizes the measurements.

Methods

The main activity for preparing the time recording is selecting the work task, work process, and work method. Workstation 4 (Figure 30) of the LEAD Factory was chosen for this purpose. A detailed description of the work to be carried out at this station can be found in Appendix A, which is used as work instruction for the LEAD Factory users. The whole process is divided into 23 small tasks that have to be processed one after the other. In the beginning, the scooter's handlebar is assembled from different parts. For this purpose, the two handlebars and the associated caps are mounted on the T-piece of the handlebar. Furthermore, the assembly of the kickstand on the frame of the scooter is done with the help of a wrench and a drill. In the last work step, the rear wheel and axle are assembled and screwed. A punch is used to position the tire to allow the insertion of the shaft correctly. If necessary, a hammer can also be used. A drill is used to screw the axle, and an allen key is used as a counterpart. All of these work steps take a total of around 195 seconds.

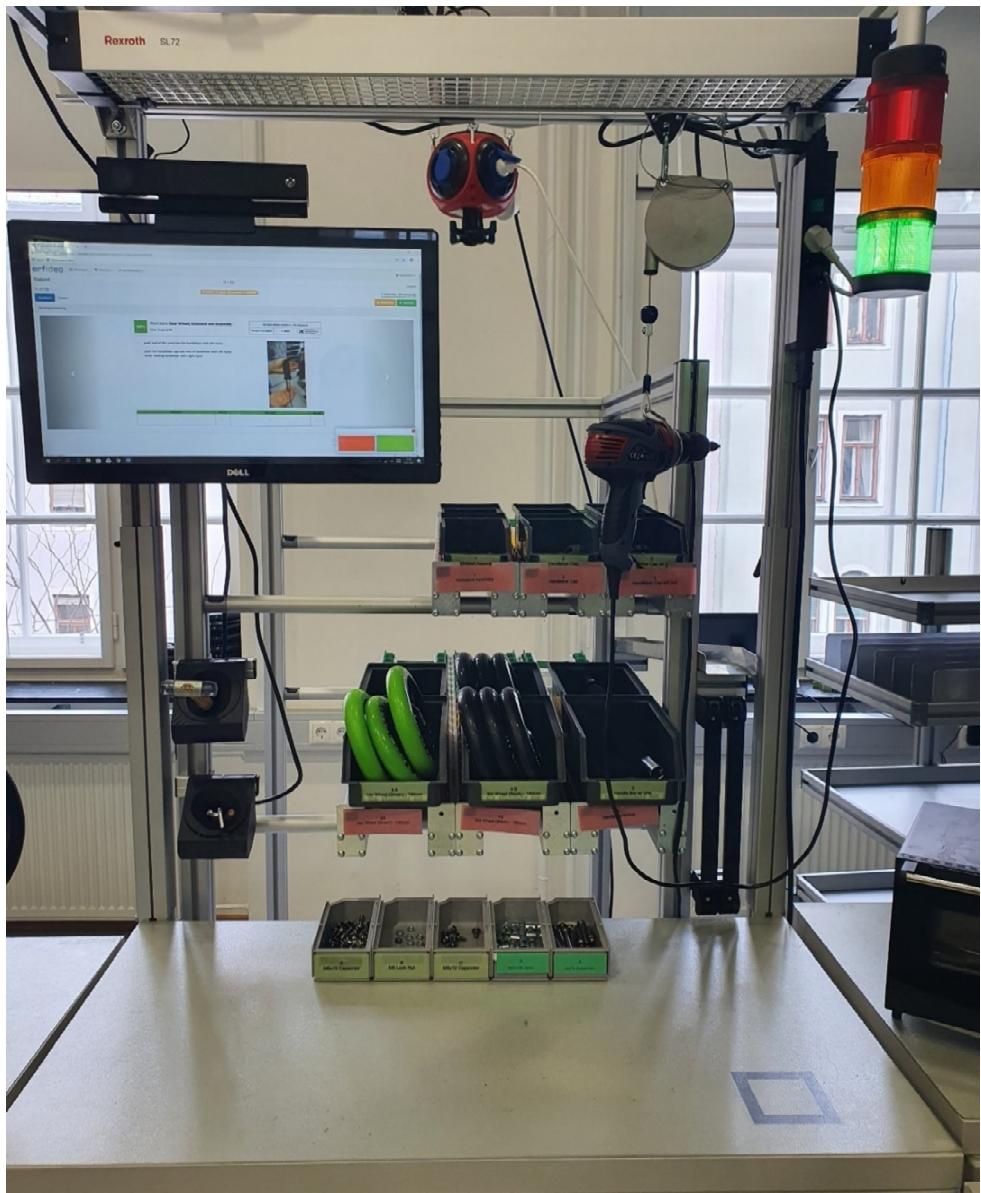


Figure 30. Workstation 4 of the LEAD Factory

Methods

In addition to these work steps, the workstation offers other options. If a new scooter's assembly process is started, the RFID tag attached to the scooter must be read via the existing RFID system. This is used to track the scooter along the process chain and identify the scooter variant, which must be observed at this workstation. Depending on the variant, two different colored tire types are available at the work station. The other RFID system functions were already explained in the previous chapter. Besides, the work instructions as shown in the appendix are displayed on a screen. Using a camera system with gesture control, the just needed work instructions can be displayed without wasting unnecessary working time.

In the next step, the measuring points for a time recording must be defined. These must be activities or states that can be recorded by the existing sensors. This essentially concerns tracking the start time via the RFID system and monitoring the 11 items available at the workplace and the maximum of 5 tools used. In the case of items, the removal from the material boxes can be recorded using position sensors. For this purpose, a sensor must be attached to each row of material boxes, i.e., in this case, three position sensors are required to cover all material boxes. Besides, tools can be registered when removing them from the corresponding tool compartments or starting position, as well as returning them to the starting position. Different sensors are used for this purpose. Wrench, punch, and allen key are stored in the same tool compartment, and their design makes them suitable for measurements using an induction sensor. By correctly positioning the induction sensor on the relevant compartment, all three tools can be precisely identified and tracked. The drill can be tracked via the existing energy monitoring system, as the energy required in the system is higher when it is in use. Also, a PMD Profiler is available, which monitors the rear wheel's correct assembly, including the axle at the end of the process. Figure 31 and Figure 32 show the installation of the sensors on the concerned workstation.



Figure 31. Installation of sensors at the workstation

Methods

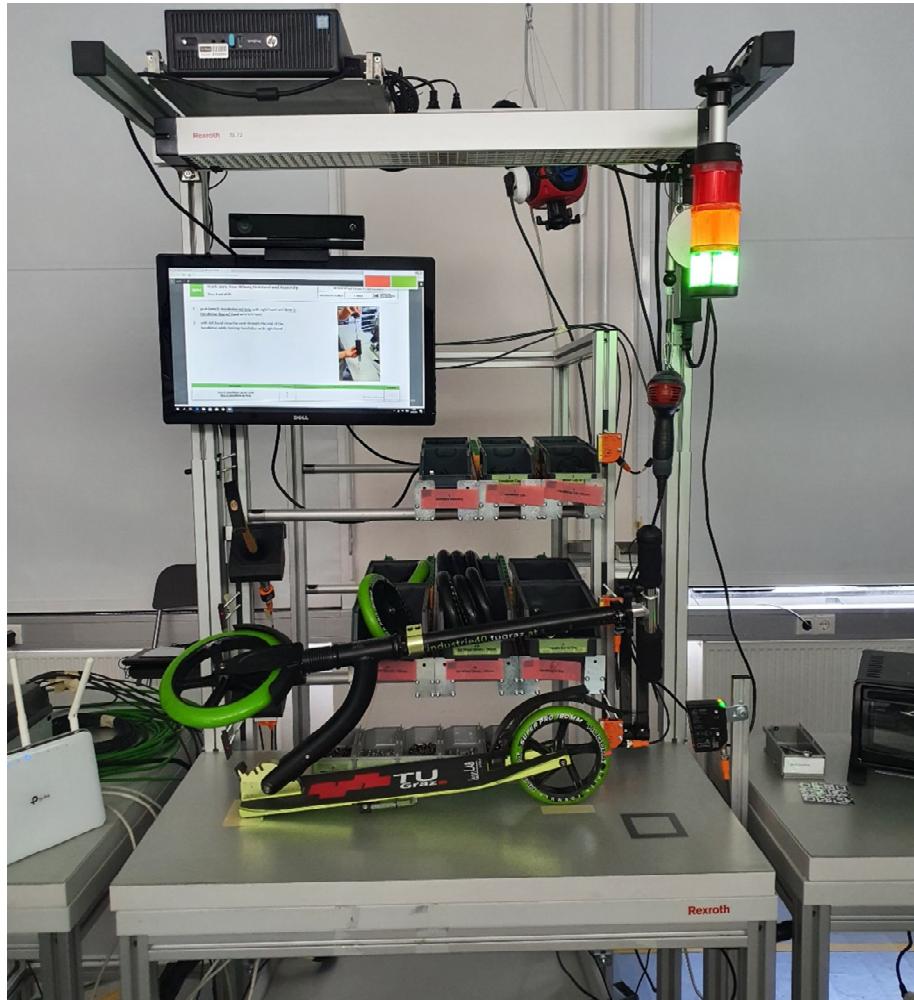


Figure 32. Installation of sensors at the workstation

The activities mentioned above can be observed every time the scooter is assembled. In addition, there are also non-cyclical activities that should be tracked and analyzed. This includes the use of the hammer, which is located in a separate tool compartment and can be detected via a vibration sensor concerning removal and return. This is possible due to the different vibration levels in both actions. Nevertheless, a detection via a capacitive sensor, which was not available, would be more suitable concerning operational safety. Other non-cyclical activities can be monitored via the RFID system. This includes the sorting out of empty material boxes as well as defective items that can no longer be used. The measuring points along the work process, according to Table 4, are the result of these possibilities. All these activities are registered in real-time and can be visualized. To carry out long-term time analyses, the process is divided into different process states of approximately the same length. This makes it possible to calculate and display the times for these states in the long term to derive trends from them. Besides, the usage times of the tools can be calculated. According to the table, the process is divided into six individual states. For this purpose, an activity has to be defined as the start point and another activity as the endpoint of the state. This means that the state's endpoint is also the next state's start point, which has to be considered, especially in the later

Methods

implementation. With this concept, the tools' usage times and the individual process step times can be viewed.

Table 4. Measuring points along the work process

| Number | Activity | Item / Tool | Process state |
|--------|----------|-----------------------------------|---------------|
| 1 | Start | RFID | 1 |
| 2 | Grab | Item 5: Handlebar w/ Grip | |
| 3 | Grab | Item 3: Handlebar Cap w/ Cord | |
| 4 | Grab | Item 5: Handlebar w/ Grip | |
| 5 | Grab | Item 2: Handlebar Cap | 2 |
| 6 | Grab | Item 1: Kickstand | |
| 7 | Grab | Item A: M6x18 Capscrew | 3 |
| 8 | Grab | Item B: M6 Lock Nut | |
| 9 | Grab | Wrench | |
| 10 | Grab | Drill | |
| 11 | Return | Drill | |
| 12 | Return | Wrench | 4 |
| 13 | Grab | Item E: Ø6x70 Shoulder Bolt | |
| 14 | Grab | Item D: Ø8x15 Spacer | |
| 15 | Grab | Item 4B Std Wheel (Black) – 180mm | |
| 16 | Grab | Item D: Ø8x15 Spacer | 5 |
| 17 | Grab | Punch | |
| 18 | Return | Punch | |
| 19 | Grab | Item C: M6x12 Capscrew | 6 |
| 20 | Grab | Allen key | |
| 21 | Grab | Drill | |
| 22 | Return | Drill | |
| 23 | Return | Allen key | |
| 24 | Control | Profile axle and tires | |

Methods

These measuring points must now be assigned to the individual sensors or sources. The individual signals must be processed so that only specific values for the respective activity remain or result. For this purpose, tags are assigned for each measuring point, i.e., activity. For example, if the wrench is removed, the induction sensor's processed signal shows a value of "21" at that time. Table 5 shows an overview of all signals, their origin, and the assigned tags. All signals processed in this way must be further processed so that at the end, a common data set or signal of all measuring points is created, in which each activity can be clearly assigned. Thus, this common data set shows all the activities collected along the time component and can be used for further processing and analysis.

Table 5. Overview of all activities, their assigned tags, sensors, and connections

| Tools | Activity | Tag | Sensor | Connection |
|-------------------------------|----------|--------------|-------------------|------------|
| Wrench | Grab | 21 | Induction 1 | X07 |
| | Return | -21 | Induction 1 | X07 |
| Punch | Grab | 22 | Induction 1 | X07 |
| | Return | -22 | Induction 1 | X07 |
| Allen key | Grab | 23 | Induction 1 | X07 |
| | Return | -23 | Induction 1 | X07 |
| Hammer | Grab | 24 | Vibration 1 | X08 |
| | Return | -24 | Vibration 1 | X08 |
| Drill | Grab | 25 | Energy monitoring | SQL |
| | Return | -25 | Energy monitoring | SQL |
| Material | Activity | Tag | Sensor | Connection |
| Item 1: Kickstand | Grab | 1 | Position sensor 1 | X01 |
| Item 2: Handlebar Cap | Grab | 2 | Position sensor 1 | X01 |
| Item 3: Handlebar Cap w/ Cord | Grab | 3 | Position sensor 1 | X01 |
| Item 4A Std Wheel (Green) | Grab | 4 | Position sensor 2 | X02 |
| Item 4B Std Wheel (Black) | Grab | 6 | Position sensor 2 | X02 |
| Item 5: Handlebar w/ Grip | Grab | 5 | Position sensor 2 | X02 |
| Item A: M6x18 Capscrew | Grab | 7 | Position sensor 3 | X03 |
| Item B: M6 Lock Nut | Grab | 8 | Position sensor 3 | X03 |
| Item C: M6x12 Capscrew | Grab | 9 | Position sensor 3 | X03 |
| Item D: Ø8x15 Spacer | Grab | 10 | Position sensor 3 | X03 |
| Item E: Ø6x70 Shoulder Bolt | Grab | 11 | Position sensor 3 | X03 |
| Other Signals | Tag | Sensor | Connection | |
| New scooter | 31 | RFID | SQL | |
| Broken part | 32 | RFID | SQL | |
| Empty shelf | 33 | RFID | SQL | |
| Profile axle and tires | 41 | PMD-Profiler | | X06 |

Based on this definition of the used signals, the data processing and visualization architecture can be worked out in the following chapter.

3.3 Software architecture design

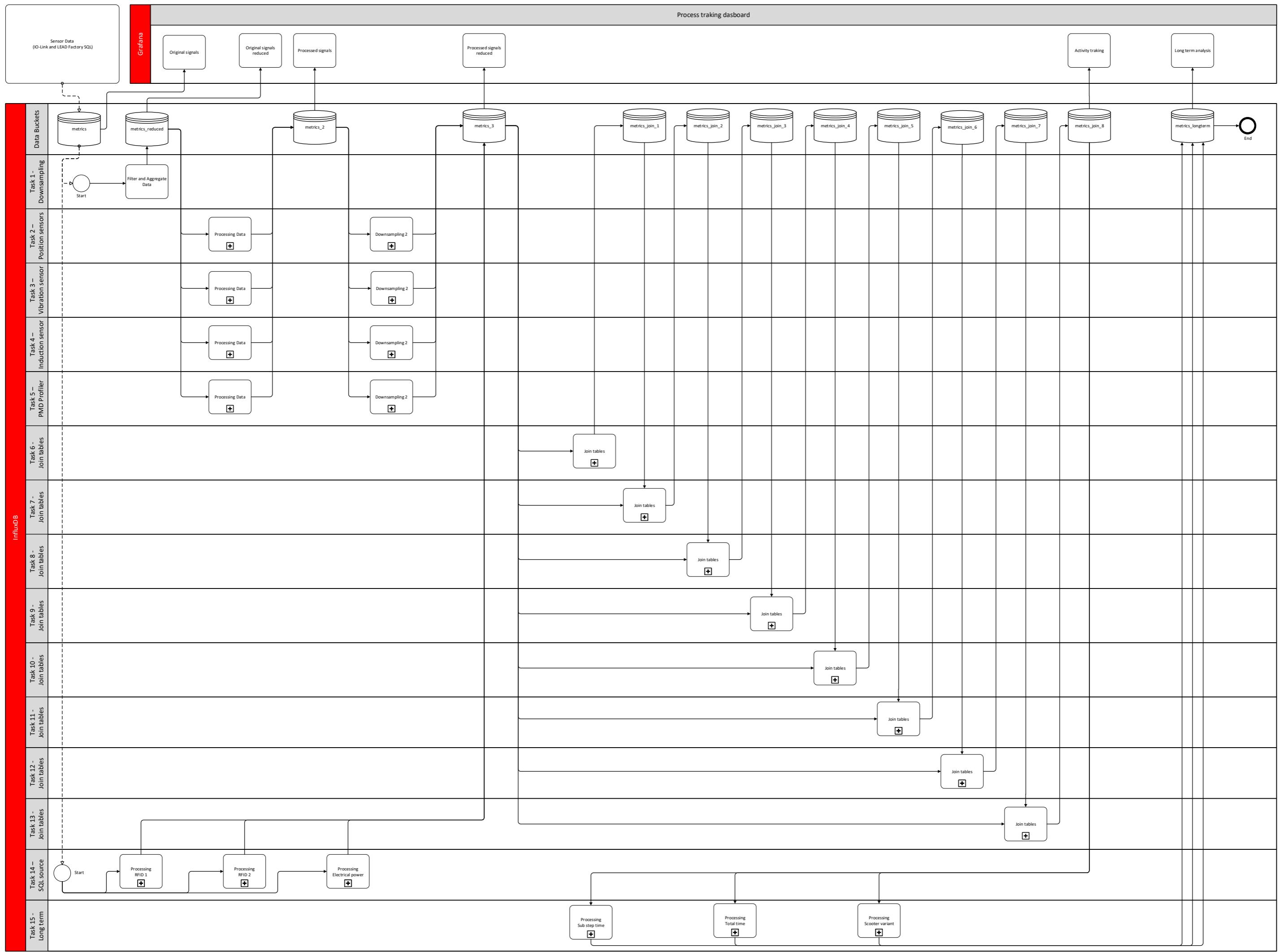
In the previous chapter, the goal of data processing was explained; at the end, all processed signals should be able to be represented in a data set. A large number of different process steps are necessary for this. For the first time, it is needed to differentiate between the two software variants in terms of their implementation, as they differ significantly due to the different ways in which they function. The graphic notation BPMN is used to visualize the entire process of data processing and preparation. This makes it possible to present the overall process as abstractly as possible. Accordingly, only the rough structure is considered. The actual data processing of the individual signals is explained in Chapter 4.

3.3.1 Software stack 1

The database of influxdata is used in this software stack. Here, the BPMN essentially describes the data stream in the database and its transfer to the Grafana visualization program. The here shown individual process steps actually consist of many other tasks and decisions. If these were integrated into the BPMN, the data stream would be challenging to understand due to its large volume. Rather, the BPMN is intended to provide an overview. The following paragraph describes the BPMN, which is shown on the next page, in more detail.

In the first section of the BPMN, the dashboard visualized via Grafana is shown in an abstract form. The original raw data is located in the upper left corner. The database and its tasks take up the most significant BPMN part. Only the first row of the database shows the existing data buckets.

The raw data is read in via the database system with so-called telegraphs and stored in the bucket *metrics*. This bucket provides the data for tasks 1 and 14. In task 1, the IO-Link sensor system's raw data is downsampled in a first step and then saved in a different bucket called *metrics_reduced*. The raw data is automatically deleted after 12 hours to preserve storage capacity. The original sensor data in the *metrics* bucket and the reduced data in the *metrics_reduced* bucket are transferred via an interface to Grafana for visualization. The reduced data is continuously called up by tasks 2 to 5. The signals from the position, vibration, and induction sensors, as well as the PMD Profiler, are processed in separate tasks to form a signal that only outputs the specific tag value in the event of activity. Otherwise, the value zero is output. It is precisely here that the BPMN is very abstract since this process, in reality, consists of a large number of steps. After processing, the data is stored in the *metrics_2* bucket. This data is also passed on to Grafana to be able to display the originally processed signals on the Grafana dashboard. The same data is then reused in the same tasks to sample it down again without losing important information.



Methods

Intermediate storage of the data in the bucket *metrics_2* is necessary due to the program logic, as these have to be read in again within the task. The result of the downsampling is saved in the bucket *metrics_3*. This bucket houses all processed and reduced individual signals. The bucket also has an interface to Grafana. In parallel to the processing of the IO-Link sensor data described above, the data from the LEAD Factory's SQL database is read in and converted into a correct format for influx. In task 14, the required signals are filtered and also processed. The RFID data is used for the activities of new scooters at the workstation, defective components, and empty material containers, as well as energy measurement at the station. The process of downsampling is not necessary for these components, as the number of measuring points is significantly lower than for the time series data of the IO-Link system. The resulting data is also stored in bucket *metrics_3*.

This essentially completes the primary processing of the original data. Each sensor's individual data set must be merged into an overall data set. This is only possible with a tremendous amount of resources in terms of the database system since only two sensor data sets can be merged per task. These amalgamations require comparatively high computing power due to their function. This is also negatively influenced due to the high number of necessary tasks. Tasks 6 to 13 execute these joins, whereby in task 6, the data sets from position sensors 1 and 2 are retrieved, merged, and stored in the bucket *metrics_join_1*. In task 7, this previously created data set is called up again, and a further data set is added. The following tasks work in the same way until all data sets have been merged into one resulting data set. This also requires a large number of buckets since the newly created data sets have to be stored in a single bucket. The final data set is saved in the *metrics_join_8* bucket and transferred to Grafana in this form. Task 15 uses this data to analyze and prepare long-term data. The times of sub-steps, total time, or even scooter variants are considered and then saved in the *metrics_longterm* bucket and transferred to Grafana. This bucket can save the data for up to 2 years. Especially with long-term analysis, the data processing does not only have to occur via an influx task. A further step in Grafana is necessary. This circumstance is related to the structure and functionality of influx.

The tasks do not run strictly linearly, one after the other. They are continuously initialized at a defined frequency (e.g., 1/s) and consider a limited period's measured values (e.g., past minute). This means that many of the tasks run parallel to one another, whereby it is essential to ensure that the tasks are evenly distributed concerning system load. Due to the considered defined periods, no time measurements can be made in influx since distances can only be measured during this period. These data can only be prepared by processing the relevant measurement data and making them available in a separate data set. Grafana can then view and analyze this data over a more extended period of time. It should also be noted that due to the architecture, the process times of the individual tasks and their cyclical initialization add up for each measured value. In the case of complex processing and a large number of process steps, the processed measurement

data can no longer be displayed in real-time. Care must be taken to keep the number of tasks and the complexity of the queries as low as possible and the frequency of the tasks as high as possible. These are competing goals; a higher frequency of the tasks shortens the maximum time in which a measured value has to wait for processing, but at the same time increases the load on the system and leads to longer process times for the tasks. Because of this, a working middle ground must be chosen. Compared to the needed architecture for time recording via InfluxDB, TimescaleDB and Node-red⁶ are used for that purpose in the next chapter.

3.3.2 Software stack 2

Software stack two only uses TimescaleDB as a database system. The actual data processing takes place with the software node-red. Therefore, the entire implementation is essentially based on the components shown in Figure 33. The sensor data is collected via the sensorconnect tool from ia: and distributed in the factorycube via an MQTT broker. The data is sent to node-red, processed, and then sent back to the broker. This broker forwards the data to a server on which the database and the visualization software are hosted.

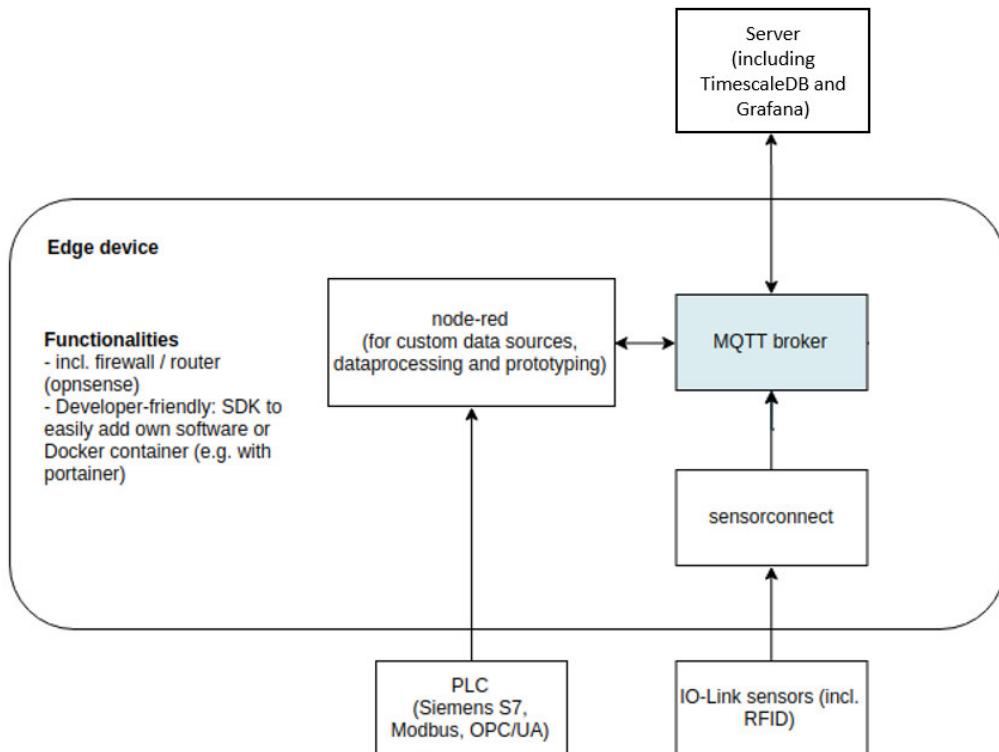


Figure 33. Relevant architecture of the software stack

⁶ Node-RED is a graphical development tool for wiring together devices in the field of the Internet of Things. “It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.” [74]

Methods

To display the data processing architecture in node-red, no BPMN is required as the graphical programming interface is already self-explanatory. Figure 34 shows the main architecture implemented in node-red. Essentially, there are three different components: data sources (output on the right-hand side), further processing (output on both sides), and output (output on the left). Distributors 1 and 2 are only used to distribute the data streams. All blocks shown in red are subflows, i.e., a large number of functions are stored in these blocks. By dividing it into subflows, the main flow shown here becomes much clearer.

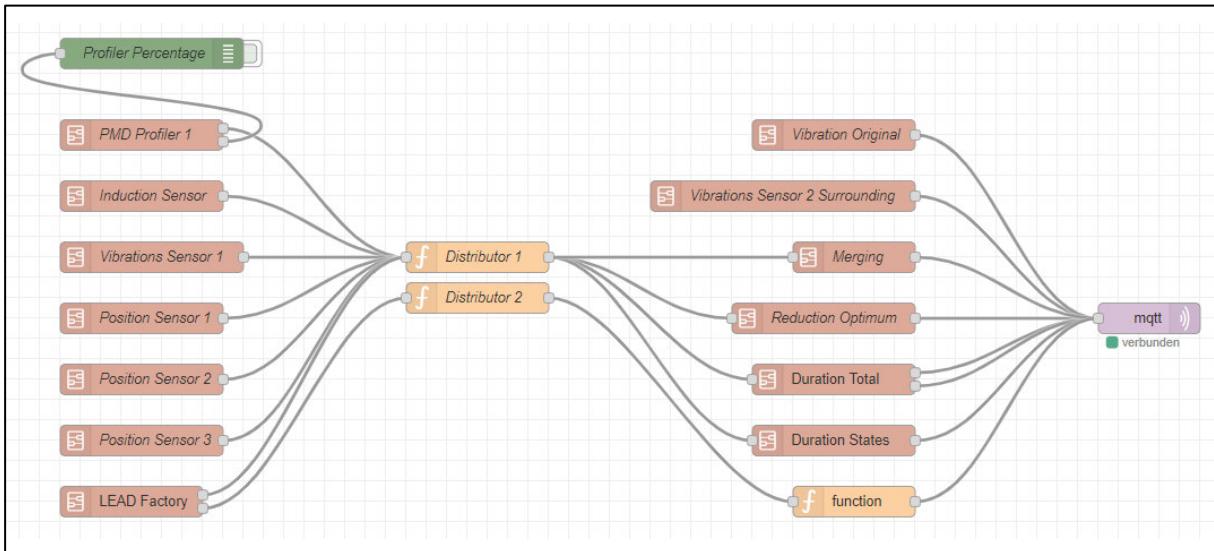


Figure 34. Data processing architecture in node-red for software stack 2

The data sources on the left side of the flow are connected to distributors 1 and 2. In the data source subflows, the respective sensor data are read in, processed, downsampled, and finally, the respective tag values are output. Besides, the subflows “PMD Profiler 1” and “LEAD Factory” provide additional information at a further output. This will be explained in more detail in Chapter 4. Distributor 1 forwards the data to various further data processing, such as merging the measured values, reducing the amount of data, or different time measurement flows. This processed data is forwarded to the server via an “MQTT” output. Also, data from another vibration sensor attached to the LEAD Factory for monitoring the workplace is transmitted via the output.

The data is only saved in the Timescale database when it is transferred via the output module. Due to the way node-red works, no intermediate storage of the data is necessary compared to InfluxDB. The MQTT network protocol is used for data input and output. So-called topics are used to identify the individual data transmitted to the server. These are for inputs in the format “ia/raw/<transmitterID>/<gatewaySerialNumber>/<port-Number>/<IOLinkSensorID>“ and for outputs in the format “ia/<customerID>/<location>/<AssetID>/processValue”. Only the input of the LEAD Factory data is not implemented via MQTT. The data is queried directly from the Microsoft SQL database.

All data stored in the TimescaleDB is forwarded to Grafana and can be differentiated by the different topics. Further processing of the data in Grafana is not necessary.

3.4 Summary

In this chapter, the experimental setup of the Factorycube concerning hardware and software was explained. The differences between the two used software stacks in this work were considered. Besides, the existing sensors of the IO-Link system and the LEAD Factory were specified in more detail in the course of the setup. In this way, the functionality of the various sensors could be displayed.

Another section of this chapter dealt with the preparation of the time recording. This includes the definition of the time recording reason and the recording type. In addition, the work process to be examined was explained concerning the work steps to be carried out, and the working environment of the workstation was considered. Based on the selected activity, possible measuring points for time recording were defined, and the possible sensors for these points were selected. This information could be presented clearly in tables. Also, initial assumptions and requirements for the implementation phase could be made, such as the assignment of interfaces, the assignment of tags, and process states' definition.

To be able to implement the time recording, a data processing architecture had to be designed. This had to be carried out separately for the two types of used software. It should be noted here that the two setups' hardware cannot be differentiated from the outside. Rather, the visualization in Grafana cannot be distinguished by the user. In contrast to this, however, the process and the internal data processing architecture differ significantly in their complexity. A closer comparison of the two systems is carried out in Chapter 4. The individual data processing steps and their visualization will also be considered in more detail.

4 Results

In the previous chapter, the two software stacks' required software architectures to carry out a digital time structure analysis on a manual workstation could be developed and described. The abstract building blocks contained therein, such as "data processing", are to be examined in more detail in this chapter. In particular, their functionality and selection should be discussed. Due to the different architectures, the specific implementation within the two software stacks is dealt with separately in this chapter. Only the visualization is considered in a separate chapter due to its overlap. If the waterfall model is reviewed again, the following subsections deal with the implementation in the form of programming, tests, and the systems' operation. To be able to make a statement about the evaluation of the two used systems, they should be compared with each other after their implementation. This serves as the basis for answering research question 2 in Chapter 5.

In addition, possible teaching options are evaluated, and a training concept is developed. Since software stack 2 with TimescaleDB and Node-red will remain on the factorycube at the end of the work, teaching options and training concepts will be created based on this setup.

4.1 Implementation with software-stack 1

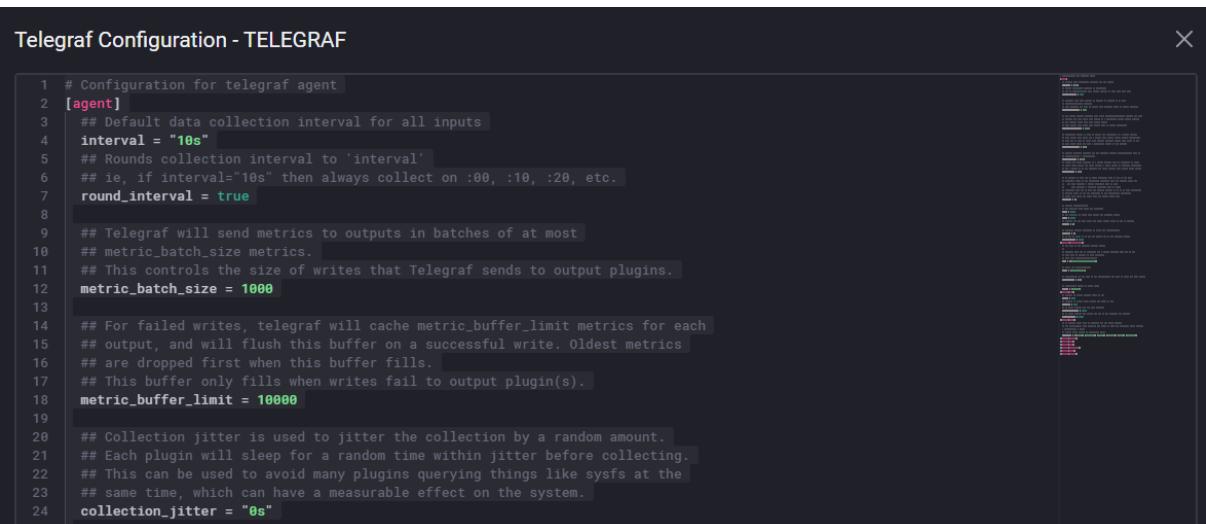
In the following, the implementation using InfluxDB will be explained. With many iteration loops within these steps, the implementation in development can be continuously improved. The implementation is primarily considered concerning data collection and subsequent processing.

4.1.1 Implementation of data collection

The data collection via InfluxDB takes place in two different ways. The IO-Link system's data is read in via a telegraph agent, whereas the SQL data of the RFID and energy monitoring system is saved via an influx task. The Telegraf agent must be configured at the beginning. Three areas are configured, the agent itself, outputs, and inputs. The agent area offers many configuration options, of which those used in this example will be briefly explained. The configuration takes place by entering the configuration in a source code. First, a default data collection interval for all inputs must be made, which was defined to be one second. Besides, the round interval function was activated, which converts the desired interval into interval limits. The collected data is not transmitted individually but in batches. To do so, the maximum batch size must be defined, which was set to the value 1000. This value was taken from the default settings. The metric

Results

buffer limit is chosen based on the maximum batch size. The Telegraf agent temporarily stores this number of measured values until the following output of a batch and thus serves as a buffer. With this function, the metric buffer limit must be a multiple of the batch size. The lower limit is twice the batch size. In our case, ten times the batch size was chosen. In addition to the data query interval, an interval for outputting the data must be defined. This so-called flush interval must be more significant than the read-in interval. Otherwise, no data can be transferred, which is why a second was also selected for the flush interval. Jitter can be used for both intervals, a collection jitter, and a flush jitter. This means that the intervals are increased randomly through the use of jitter. With a jitter of 500ms, the original interval of 1s can now be between 1 and 1.5 seconds. Using jitter, large write or read spikes can be prevented when using several telegraphs. However, it should be noted that the input interval plus jitter results in the minimum flush interval. Because only one telegraph is used, there is no jitter at this point. Finally, it was defined as precision milliseconds. Further standard settings could be adopted.



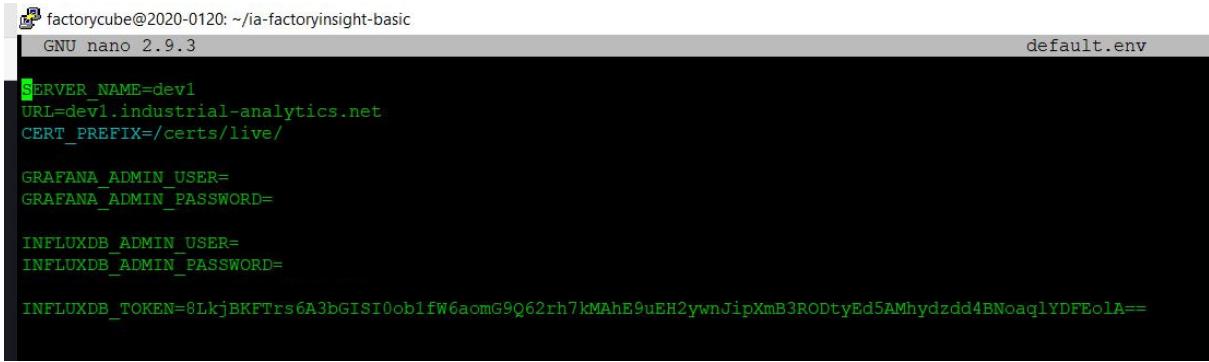
```
1 # Configuration for telegraf agent
2 [agent]
3 ## Default data collection interval for all inputs
4 interval = "10s"
5 ## Rounds collection interval to 'interval'
6 ## ie, if interval="10s" then always collect on :00, :10, :20, etc.
7 round_interval = true
8
9 ## Telegraf will send metrics to outputs in batches of at most
10 ## metric_batch_size metrics.
11 ## This controls the size of writes that Telegraf sends to output plugins.
12 metric_batch_size = 1000
13
14 ## For failed writes, telegraf will cache metric_buffer_limit metrics for each
15 ## output, and will flush this buffer on a successful write. Oldest metrics
16 ## are dropped first when this buffer fills.
17 ## This buffer only fills when writes fail to output plugin(s).
18 metric_buffer_limit = 10000
19
20 ## Collection jitter is used to jitter the collection by a random amount.
21 ## Each plugin will sleep for a random time within jitter before collecting.
22 ## This can be used to avoid many plugins querying things like sysfs at the
23 ## same time, which can have a measurable effect on the system.
24 collection_jitter = "0s"
```

Figure 35. Configuration of the telegraf agent

In addition to the agent level, as shown in Figure 35, the output level of the telegraf must be configured. Various plugins are available for the different influx versions. At this point, the plugin outputs.influxdb_v2 was selected, which writes the data in the flush interval into the influx database. To be able to, the organization in influx (“ia”) and the target bucket (“metrics”) must be selected. The database itself is connected via the IP address 172.16.20.3:8080, which must also be stored. Finally, an access token must be stored in the configuration and the database. The data transfer is approved by the agreement of the token in the database and the configuration of the telegraf. If these do not match, no data can be transferred.

Similarly, a plugin is also used for the input of the data, whereby it is the plugin inputs.docker. This plugin also requires a token, which is stored in the Docker system itself, as shown in Figure 36. This token can be changed in the .env file via an ssh connection (secure shell) via a putty client.

Results



```
factorycube@2020-0120: ~/ia-factoryinsight-basic
GNU nano 2.9.3                                     default.env

$ SERVER_NAME=dev1
$ URL=dev1.industrial-analytics.net
$ CERT_PREFIX=/certs/live/

$ GRAFANA_ADMIN_USER=
$ GRAFANA_ADMIN_PASSWORD=

$ INFLUXDB_ADMIN_USER=
$ INFLUXDB_ADMIN_PASSWORD=

$ INFLUXDB_TOKEN=8LkjBKFTrs6A3bG1SI0ob1fW6aomG9Q62rh7kMAhE9uEH2ywnJipXmB3RODtyEd5AMhydzdd4BNoaqlYDFEo1A==
```

Figure 36. Token for the influx telegraf (shown via the putty client)

In addition to the IO-Link data, the LEAD Factory's existing data must be read in. This activity is done via a task, shown in Figure 58 (Appendix). For this purpose, the factorycube is integrated into the LEAD Factory network using wifi. The necessary settings can be made in the router of the factorycube. A connection via wifi or lan must be established with an end device to access the router. It can then be accessed via IP 172.16.20.1. After successfully integrating the factorycube into the wifi, the Microsoft SQL Server 2014 under LEANLAB\ERFVIDEO as the LEAD Factory data storage can be accessed by task 14 using source code or query. The data thus read in is then processed by Influx concerning the time stamp format and filtered based on the required data. Each RFID entry has the following information: ID of the database entry (ID), the ID of the scanned object (ProductID), read point or workstation ID (ReadPointID), read point or workstation name (ReadPointName), scanning time of objects (TimeStampEvent), the status of the scanned object (Status), name of the scanned object (ProductName) and article number of the scanned object (ArticelNumber). For the use of the data in the context of this work, only TimeStampEvent, ReadPointName, ProductName, and Status are used since all necessary knowledge can be obtained from this information. Following these activities, the received data is saved in the bucket *metrics*. This bucket thus houses all the raw data that arises.

4.1.2 Implementation of data processing

Influx tasks are called cyclically by the database system, e.g., every second, and run through the queries (source code) defined in the tasks for each data line in a specified time window. Due to every single line's query, it is impossible to work with variables in queries. This means that, for example, counter variables cannot be used. Processing the data, in particular activities such as counting entries or calculating times, is therefore difficult. Suitable approaches must be found for this to be able to circumvent this limitation. The tasks and contents of the individual created tasks will be explained in the following. For this purpose, the source code of the tasks was visualized using flowcharts, which can be found under Appendix B. The source code itself is not shown at this point for better understanding.

Results

Task 1 (Appendix B; Figure 50)

This task covers the first step in downsampling the data. The number of measuring points of the raw data is reduced to guarantee better performance in terms of processing speed. This is necessary because of the large volume of data. Each IO-Link sensor generates around 60,000 measured values per hour. In a first step, the addressed bucket must be defined, from which the data for performing the task originate. The data is then filtered according to the required data, as the Telegraf agent also records system data such as CPU utilization and storage capacity. With this filtering, the field column is used, in which information about the sensors can be found. “X02_primaryValue” would be, for example, the first sensor value table of position sensor number 2. For redundancy reasons, the signals are stored twice in the database by the telegraf, so there is also an “X02_secondaryValue”. A new value is assigned to the measurement column to assign the filtered data clearly. Besides, unnecessary columns such as host details are discarded for data reduction reasons. Averages are calculated over a defined time window in the task’s essential step. A time window of $t = 200$ ms was selected, i.e., five measured values are output per second. This results in a reduction in the amount of data by around 70%. However, the time window should not be selected too large since otherwise important signals will be erroneously smoothed and thus cannot be clearly assigned in further processing. After intensive tests, the value of 200 ms is the limit value at which all critical signals are retained, but at the same time, a high data reduction takes place. A further reduction is carried out in a later step, taking this problem into account under tasks 2, 3, 4, and 5. The process described here, “aggregate window with mean,” was shown as a sub-process in the flowchart. Essentially, it contains the implementation of mathematical formulas (see Chapter 2.8) with the help of source code. Finally, the reduced data is saved again in a bucket.

Task 2 (Appendix B; Figure 51 and Figure 52)

Task 2 processes the already reduced signals from the position sensors into a signal that, according to Table 5, shows the individual material compartments’ activities. The position sensors provide distance values and should be assigned to separate compartments. However, this creates several problems. Each container is not only assigned one distance value but rather a range of values corresponding to the compartment width, i.e., compartment one, for example, is assigned the value range 20 to 35 of the position sensor 1. In addition to this difficulty, ambiguous values can arise. If compartment one is reached, a value outside the range of values can briefly occur when approximating the sensor value to the actual value. It is to be avoided that these “wrong” values are assigned to other material boxes and thus falsify the time measurement function. Implementing the precise compartment detection by taking this problem into account will now be described.

Results

At the beginning of the task, the necessary sensor data is loaded from the *metrics_reduced* bucket and filtered, whereby all three position sensors can be processed simultaneously. The contained data values are checked to determine whether they are in a defined value range of a material compartment. It should be noted that the relevant sensor must permanently be assigned to the specified value ranges. Otherwise, an assignment to the respective subject is not possible. For example, a range of values could simultaneously affect a box in rows one, two, or three. If a checked measured value lies in a value range, it is assigned a new value according to the formula $value = n * 10^n$. The variable n describes the number of the corresponding compartment. If a measured value is not in a value range, the value 0 is assigned. At the end of this process, each measured value is assigned either the value 0 or one of the 11 specified values. The previously mentioned “incorrect” measured values are still included. The use of an exponential function at this point makes it possible to eliminate these values later. For this purpose, the resulting measured values are smoothed using a moving average with a width of two measured values. Figure 37 shows an exemplary measurement curve before using a MA (blue) and the same curve after using a MA (orange). In the blue measurement curve, the critical incorrect value was marked in red (in this example, 200). This value would be wrongly assigned to compartment number 2 without any activity taking place there. After completing an MA, this value is eliminated. This results in new values (100, 1600, and 1500), which cannot be incorrectly assigned to a material compartment.

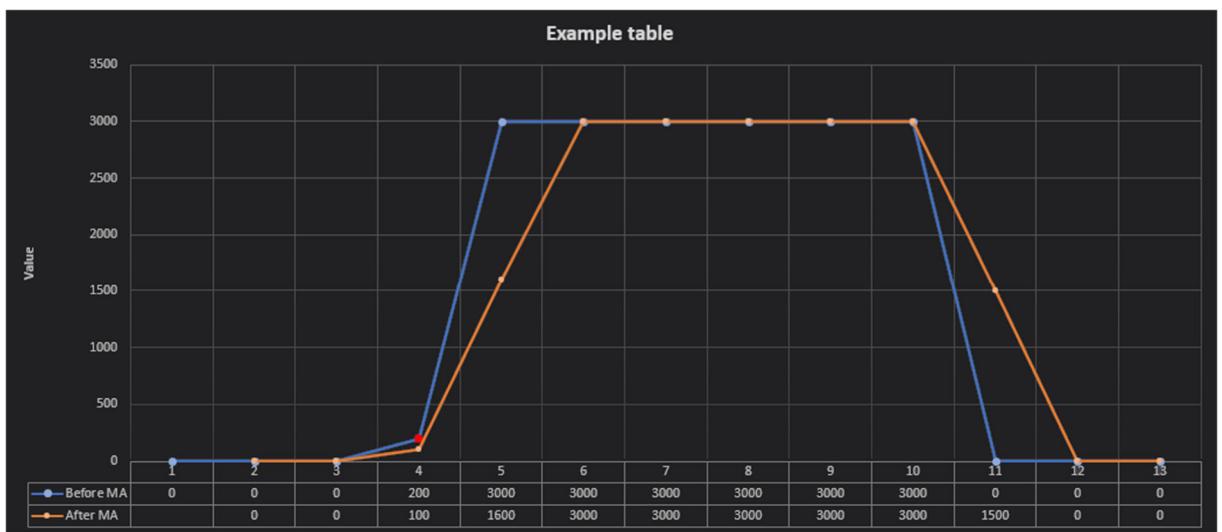


Figure 37. Example table to position sensor data processing

After this process steps’ results have been temporarily stored in a bucket, the unique values are assigned to the tags defined in Table 5. The remaining useless values (here 100, 1600, and 1500) are eliminated. This creates a measurement curve with a straightforward assignment. The only disadvantage of this procedure is that a correct value is “wasted” by the MA at the beginning and the end of the respective measured values. This fact slightly reduces the accuracy and possible reactivity in the case of very

Results

short events. In a further step, the resulting plateaus are converted into positive edges to receive only a defined single signal. For this purpose, the difference between two neighboring measured values is calculated, and the negative component is then set to the value 0. As in the flow diagram for task 1, mathematical processes such as moving average and compute difference were shown in a simplified form as subprocesses in the flow diagrams for task 2.

Task 3 (Appendix B; Figure 53)

In the next task, the vibration sensor signals are used to detect hammer activities in the tool holder. The vibration sensor detects the slightest vibrations so that the output of the sensor values constantly fluctuates around the trend component. These fluctuations are smoothed out relatively well by downsampling in Task 1. However, the rashes for removing the hammer or putting it back are much more significant. As shown in Figure 38, the rest of the signal looks like a straight line compared to an activity's strong amplitude. This fact is due to the scaling of the graphic. Nevertheless, it can be seen that the vibration values are always at least at a level of just below the value of 5000.

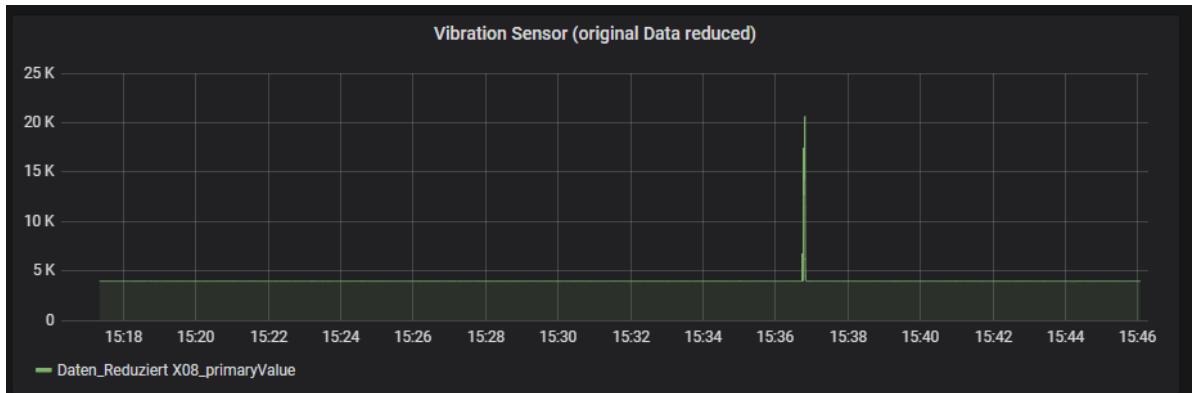


Figure 38. Example data of the vibration sensor

After the raw data has been loaded and filtered, the signal's pure changes should be made visible. For this purpose, a difference is used between the respective neighboring values, which creates a measurement curve around the 0-axis. The pure signal changes can thus be better detected. Two value ranges could be defined for the activities to be considered through repeated attempts. The value zero should be assigned to the signal under threshold one since negative signals, and small fluctuations around the 0-axis are not relevant. In the range between threshold one and threshold two, the value is set to 24 for removing. If the value is higher, it is set to -24 for return. Since the value range below is also passed through with a value above threshold 2, the value 24 would be shown in the output shortly before the value -24. To avoid this error, the signal is divided into small aggregates windows with $t = 400$ ms, and the minimum value in the window is determined and assigned to a new measuring point. Thus, in a window with the values 0, -24, and 24, only -24 remains. Tests have also shown that this system is functional with constant use but reacts sensitively to manipulations on the user's part, so using a

Results

capacitive sensor can be recommended as a suggestion for improvement. Finally, new values are assigned for columns such as measurement, and columns that are not required are deleted. The data set is saved in the bucket *metrics_2*.

Task 4 (Appendix B; Figure 54)

In another tool container at the work station, the user has three tools at his disposal, which can be traced via an induction sensor. As a basis for this, each tool must be clearly assignable in the sensor's measurement curve. This means that each tool causes a specific sensor value different from the other when it is removed and returned. If only the tool container is used, the devices can be stored in many different positions. It would not be possible to assign the sensor values. To avoid this problem, a tool insert for exact positioning was designed and implemented with a 3D printer, as shown in the middle and left in Figure 39. Thus, the tools' positioning is always the same, and the sensor values can be assigned. Additionally, the punch and the allen key are almost identical in their inductivity values, so the induction sensor must be positioned so that the two tools can still be distinguished. For this purpose, the sensor was moved in the allen key's direction. A sensor holder was designed and printed to guarantee constant measured values, which takes the positioning described above into account.



Figure 39. Tool container with tool separation and sensor holder

The loaded and filtered raw data is first divided into aggregate windows, and their averages are calculated. These averages are then rounded to whole numbers. The reason for this procedure is the occurrence of high-frequency fluctuations. In most cases, the sensor outputs a continuous line, provided there is no activity. However, there are regular fluctuations between two sensor values, i.e., the sensor value changes between two values

Results

at high frequency. This is thus suppressed. By querying the tools' specific sensor values, the values 21, 22, and 23 can be assigned to them. Otherwise, the value 0 is assumed. This creates a measurement curve with the corresponding plateaus when the tools are removed and returned. It should be noted that only one device is removed and then put back again at the same time. The plateaus become the positive edges for the removal and the negative edges for the tools' return by calculating a difference. Finally, new values are assigned for columns such as measurement, and columns that are not required are deleted. The data set is saved in the bucket *metrics_2*.

Task 5 (Appendix B; Figure 55)

The data from the PMD Profiler is processed as an additional sensor. Basically, functions that have already been used are used again. The signal, which lies in the range between 0 and 100 and indicates the correspondence of the measured profile in percent, is processed using the aggregate window with maximum value and assigned to the defined tag 41 via threshold queries. A final measurement curve with a clearly visible function results from a difference and negative values' truncation. The data set is saved in the bucket *metrics_2*.

Task 2 to 5 (Appendix B; Figure 56)

In addition to the data processing described above, a second downsampling of the data is carried out in tasks 2 to 5. Figure 40 shows an exemplary measurement curve, which can be divided into two areas, a straight component (1) and a measured value deflection (2). The measured value deflection is the component that is relevant for all processes. No data should be lost here, i.e., the resolution of this area should be kept as high as possible. On the other hand, the straight-line component is of less interest and can be significantly reduced in its number of measuring points. With this approach of dividing the data into two segments, a significant reduction in data can occur.



Figure 40. Measurement curve with two different areas

The corresponding measurement curves are called up twice in parallel and specifically processed to implement this approach in one task. In one of the two measurement curves, the straight-line component is filtered; all that remains is the deflection of the measured

Results

values. The second loaded measurement curve is significantly reduced in terms of the number of measurement points with aggregate windows $t = 3$ s and their maximum value. Besides, deflections in the values are filtered, resulting in a straight measurement curve with a small number of measurement points. These two measurement curves are then merged and result in a measurement curve that optically corresponds to the original measurement curve but has fewer data points in non-relevant measurement areas. The downsampling factor cannot be precisely defined according to the number of deflections but can be up to 13 after attempts with the existing settings. However, since each signal has to be processed separately, the high implementation effort and the associated high computing power requirements must be cited as disadvantages.

Task 14 (Appendix B; Figure 58)

As a second data source, RFID and energy measurement data from the LEAD Factory is used. This data can be divided into three data strings: RFID sensors 1 and 2, as well as energy measurement data. RFID sensor one is attached to the work surface and detects the start of a new scooter and defective components. These events can be determined by simply querying the data and assigning the resulting tags 31 or 32. Besides, it should be determined which of the 11 components is defective (tag 32). This information is entered in the column field of the respective measured value and can be used later in Grafana. The signal of the second RFID sensor, which is positioned under the work surface, is filtered and queried for empty shelves. These empty shelves can be detected and assigned with the tag 33. On the other hand, the energy measurement data is smoothed at the beginning with a moving average to be able to evaluate better the measurement curve concerning the permanent exceeding of a threshold. If this threshold is exceeded, 25 is assigned to the measured value. This creates a measurement curve with plateaus when using the drill. To obtain positive and negative edges, differences are calculated. The processed data is finally saved in bucket *metrics 3*.

Tasks 6 to 13 (Appendix B; Figure 57)

Tasks 6 to 13 combine the processed data one after the other to form an overall data set, which contains all relevant information. To be able to merge the data, the relevant data sets must be loaded and filtered. The entire layout of the data sets must then be adapted to one another to be compatible. This includes column names, number of columns, measured values format, etc. The two data sets are processed in parallel. Once these have been prepared, they can be put together. There are two different variants: Join and Union. When joining two data sets, all existing measured values are combined, i.e., two measured values with identical timestamps become one new metric value. Mathematical functions such as addition or subtraction can be used. In total, this results in an equal or even smaller number of data points for a new data set. However, the problem here is that a join only results in data points if two timestamps match. Due to the downsampling

Results

that has already been carried out, it cannot be guaranteed that this applies to all measured values. One consequence can be that a tag of an activity does not have a corresponding measured value in the other participating measurement curve and is therefore lost. The basic information in the data set would then no longer be available. On the other hand, in the case of a union, all measured values are transferred to a new data set. The sum of the data points results from the number of measurement points of the two measurement curves. In this case, no information is lost, but at the same time, the number of data points in a single measurement curve is significantly increased. Therefore, recurring downsampling at the end of the individual mergers would be recommended. This downsampling can be implemented, for example, with aggregate windows and the use of maximum values. This also prevents the redundancy of measuring points with the same timestamp and the possible inconsistency when tags and zero values come together. The most appropriate approach was chosen individually for the individual mergers due to the difficulties described here. Finally, the newly created data set is saved in the respective bucket defined according to the BPMN.

Task 15 (Appendix B; Figure 59, Figure 60, Figure 61, and Figure 62)

The resulting data set in the bucket *metrics_join_8* of the previous tasks depicts all activities from which long-term data are now generated. For this purpose, this task is also divided into three separate program sequences: detection of total time, substep time, and scooter variant detection. Besides, this task can be expanded if necessary. To determine the entire duration of a scooter installation at the workstation, the data set is filtered to the tag values initially. If available, tags 31 and 41 for the activities “new scooter” and “control profile axle and tires” are assigned the value “Duration” for the field column. The data set is then reduced to measuring points with the assigned value “Duration,” and the column measurement is renamed. This data can then be used to determine the duration through the visualization using Grafana. It should be noted that only the period from 31 to 41 and not from 41 to 31 should be calculated. In addition to the total time, the substep times for the process steps (defined in Table 4) and times for the individual tools are detected. In the beginning, all zero values are again filtered out, and the respective states are determined in two parallel sub-processes. As described in Section 3.2, the difficulty is that the end of a state is also the beginning of a new state. This does not apply to the beginning of the first state and the end of the last state; these are only assigned to one state. To determine the durations, the relevant measured values must be saved in a separate data set for the states. Figure 50 (Appendix B) shows these sub-processes broken down. States 1, 3, and 5 are assigned to the relevant tags in one path and states 2, 4, and 6 in a second path. As a result, the appropriate measured values are stored twice, i.e., once as the state’s end and the next state’s start. Figure 41 shows this process using two states with the values 1, 2, and 3 as an example. The value 2 can be found in both the first and the second table. With this procedure, the state is stored in the field column. This process results in a total of six different data sets, one data set

Results

per state. As already described, these data sets can then be evaluated concerning their duration via Grafana. It should be noted that again only the individual states' duration is calculated and not the duration between a renewed call of a state. This requirement is solved in Grafana using filters and will be described in the following chapter.

| | | | | |
|--------------|-------------------------|-------------------------|---------------|---------------|
| 0 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 1 | State_1 |
| 0 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 2 | State_1 |
| 0 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 1 | State_1 |
| 0 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 2 | State_1 |
| false | true | true | false | true |
| long | dateTime:RFC3339 | dateTime:RFC3339 | double | string |
| table | _start | _stop | _value | _field |
| 1 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 2 | State_2 |
| 1 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 2 | State_2 |
| 1 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 3 | State_2 |
| 1 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 2 | State_2 |
| 1 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 3 | State_2 |
| 1 | 2021-02-01T08:40:39Z | 2021-02-28T09:40:39Z | 2 | State_2 |

Figure 41. Representation of the grouping in different states with multiple values

For scooter variant detection, the scooter variants are also determined using the tags. If Item 4A (Wheel in Green) is installed, it belongs to variant B. When Item 4B (Wheel in Black) is used, it is assigned to variant A. The respective variant is entered in the field column and can be picked up again by Grafana at a later point in time. Another possibility to analyze the use of the different variants would be the further processing of the RFID data since the information about variants is also stored here. All information determined in this task is saved in a data set in the bucket *metrics_longterm*.

4.2 Implementation with software-stack 2

This chapter explains data processing using the second software stack with the Timescale database and the graphical programming tool node-red. Basic data processing procedures can be taken from the previous chapter.

4.2.1 Implementation of data collection

With software stack 2, the data is read differently from various sources. The resulting IO-Link sensor data is read in using the sensorconnect tool provided by ia:. This tool which is stored in a docker container, automatically finds connected ifm gateways, extracts all relevant data, and pushes the data to an MQTT broker in the highest possible data frequency. This automation significantly simplifies the process of data collection concerning the IO-Link sensors. All connected sensors that transmit data can be made visible using an MQTT explorer, as shown in Figure 42.

Results

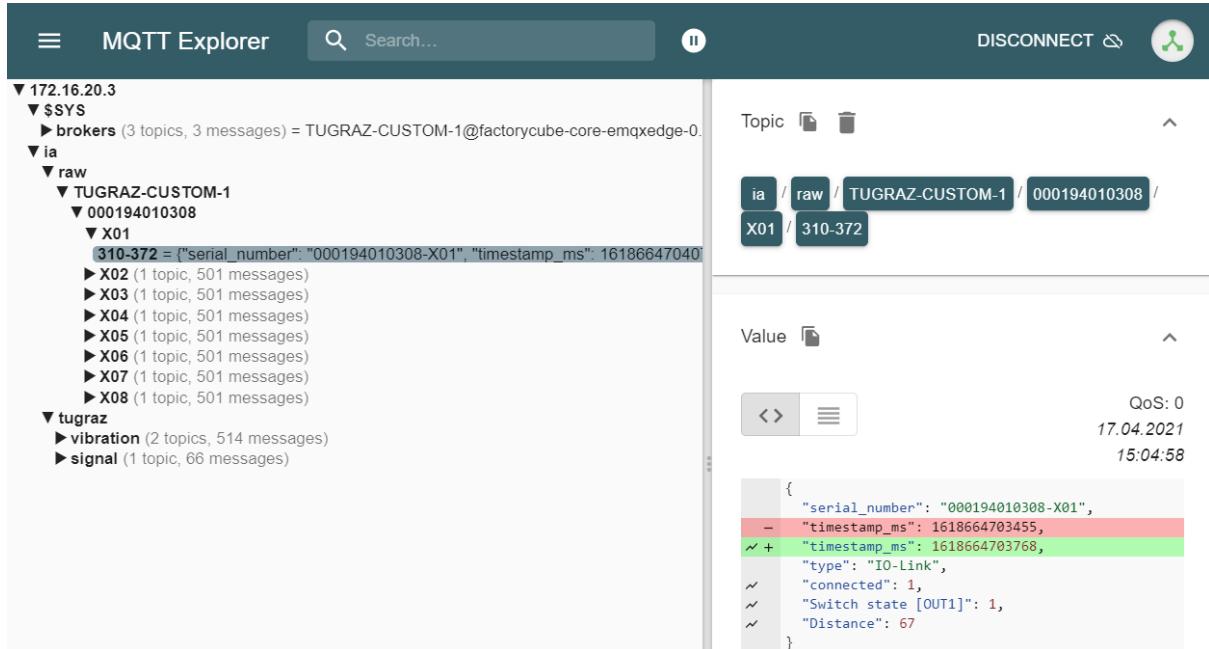


Figure 42. Showing all signals via the MQTT Explorer

In addition to the sensor signals shown under the “raw” tab, the data sent to the server is also visible under “tugraz”. The data of the respective sensors are picked up in node-red by corresponding “MQTT in” nodes. These nodes only have to be configured by specifying the server and the topic. The factorycube must be specified as the server. The respective topic in the format “ia/raw/<transmitterID>/<gatewaySerialNumber>/<portNumber>/<IOLinkSensorID>“ can be taken from the view in the MQTT Explorer. In Figure 42, this is shown as an example of a position sensor. After the configuration of the MQTT nodes, the data can be further processed in node-red. The data is output as a string with all associated information.

On the other hand, the LEAD Factory data are not read in via MQTT. This data can be queried in node-red via a special “mssql” node after connecting to the network of the learning factory, as shown in Figure 60 appendix. The node must be connected to the server of the Microsoft SQL database “LEANLAB\ERFIDEO”. Via this connection, the database of the learning factory is queried cyclically for new entries. The corresponding query is necessarily an SQL query due to the target database. The tabular data from the MSSQL database is then available in node-red in the format of arrays for further processing.

4.2.2 Implementation of data processing

The graphical implementation with node-red offers, compared to InfluxDB, extended possibilities in the data processing. This chapter is mainly intended to show the differences to the previous implementation. Node-red offers a lot of different standard nodes for implementation, which covers a wide range of functionalities. If no standard node is available for the desired functionality, the online database of node-red should first be

Results

searched for desired nodes. In this database, node-red users can make nodes that they have created themselves available to other users. Besides, special functions can be implemented in function nodes via JavaScript if no node is available for the respective application.

The data collection was already described in the previous chapter. The sensor data is transmitted to node-red as a JSON string. To be able to process this data, it must first be converted into a JavaScript object representation for each sensor, and the desired information, such as sensor values, has to be extracted. A JSON node and a self-written function are available for this purpose. The subflow “vibration original”, according to Figure 69, passes on the original raw data of the vibration sensor after the type conversion described above. Only the function has to be expanded to include the topic for identification. This subflow shows the minimum amount of data processing necessary with node-red to transfer sensor values to Grafana. The first downsampling stage, as mentioned in task 1 (influx), is carried out separately within the individual sensor data processing.

PMD-Profiler (Appendix C; Figure 64)

Processing the PMD-Profiler data, there are differences compared to Influx in the data format. After converting the data, node-red receives a hexadecimal number sequence with 40 characters. Each four characters form a message, which are transmitting specific information from the sensor. In this case, only the first four characters are required. These are extracted by a split every four characters and a throttle node, which allows every ten-character block to pass. A function converts these four characters into a decimal number. The result indicates the percentage agreement of the profile measured with the sensor. In a function, the defined threshold for triggering a signal (profile matches) is monitored, and, if necessary, the tag value or zero is assigned. The signal then passes through the nodes rbe and rising edge. The node rbe only allows values to pass if they are not equal to the previous value, i.e., the tag value 41 can only be followed by 0. The node rising edge then filters out the value 0. Ultimately, if the profile matches, only one tag value is output.

Induction Sensor (Appendix C; Figure 65)

However, the data processing of the induction sensor is almost identical to the implementation in Influx. The data processing was only supplemented by the two-stage procedure for eliminating “flawed” values already described in Task 2 (position sensor influx). The moving average contained therein can be implemented in node-red by a simple node. Only the size value of the MA has to be defined. Besides, the data processing of the induction sensor in node-red was supplemented by a rbe node at the end of the flow. This ensures that the same tag value is not incorrectly output several times in a row.

Results

Vibration Sensor (Appendix C; Figure 66)

The implementation of the vibration sensor was supplemented by aggregate windows with a sum after the formation of a difference. This makes the difference between the events hammer is removed and put back even more precise, which leads to a significant reduction in false detections. A rbe node at the end of the flow ensures that tag values are only output once.

Position Sensors (Appendix C; Figure 67)

The data processing of the position sensors could also be implemented almost identically. However, the final formation of a difference can be dispensed with a rbe node, because it was only used to generate a single edge value.

LEAD Factory (Appendix C; Figure 68)

After the MSSQL node has queried the data, the data is transferred to two functions in the form of an array. Similar to Influx Task 14, a function evaluates the data after the events “new scooter,” “broken part,” and “empty shelf” and assigns the specific tag values. The use of the electric screwdriver can unfortunately no longer be detected at this point, as a cordless screwdriver replaced it due to a defect. The attachment of a capacitive sensor could make this detection possible again in the future. In a second function, the respective components are determined for “broken part” events, and a tag value analogous to the used components from material boxes is assigned. Each of these eleven different tags is assigned a specific topic, and the data is transmitted directly to the server. This procedure makes it possible to create a histogram of the defective components in the visualization.

The subflows described above usually output the specific tag values based on the registered sensor values. In the implementation with node-red, “0” values are dispensed to keep the amount of data deliberately small for further processing. All resulting data is bundled by the node “distributor 1” and sent to the four further processing subflows. In contrast to Influx, no computationally intensive joins and unions are necessary by bundling the data. The structure of the four further processing subflows or sub-functions cannot be compared with influx due to the functionalities and options provided by node-red. The implementation with node-red has been fundamentally changed, thereby simplified and improved at the same time. These sub-functions are briefly described below.

Merging (Appendix C; Figure 71)

In this subflow, all previously mentioned data are combined, and a “0” value is added cyclically to obtain an overall signal that can be visualized. Compared to the implementation in Influx, the amount of data has decreased but can be further reduced through optimizations. This was implemented in the following subflow.

Results

Reduction Optimum (Appendix C; Figure 72)

In this subflow, the amount of data is reduced to a necessary minimum. The incoming data is distributed over three modules, filter 1 immediately outputs a “0” value. Filter 2 also outputs a “0” value, albeit with a delay of 0.5 s. In the third path, the original value is output unchanged with a delay of 0.1 s. This procedure embeds the initial value in two “0” values. This results in the absolute minimum of measuring points for correct output. Finally, the data is again provided with a specific topic.

Duration Total (Appendix C; Figure 73)

To measure the entire assembly time, only the relevant tag values for the beginning and the end of the existing data are considered. Two switch nodes ensure that only these two events are considered. Using a rbe node ensures that only start and end values can occur alternately. If a value passes this node, it is simultaneously sent to a node to measure the interval length and a join node. The node interval length uses every incoming signal as the end and start value for its time measurement and outputs the current measured value. This measured value is merged with the triggering tag value by the join node to form an array. The join is always triggered after the first incoming value within a short time window. This join makes it clear from the array whether it is the measured duration of an assembly process (start to finish) or the duration of a break between two assembly processes (end to start). A function is used to send these two time measurement values to different outputs and to assign different topics for later identification.

Duration States (Appendix C; Figure 74)

The Subflow Duration States measures the duration of the states defined according to Table 4 with the same sequence as in the subflow described above. Only the tag values have to be adjusted for each state and a unique topic assigned. It is unnecessary to measure the break time as this is not relevant.

The Scooter Variant detection carried out in Influx could be dispensed with in node-red. This function can now be carried out directly in Grafana by transforming the transmitted overall signal. By assigning topics to all of the signals above sent via MQTT, these can be clearly assigned. To access the data, the topic must begin with the customerID “ia/tugraz/” according to the ia: data model. If the data is to be output again unchanged in Grafana, the respective topic must describe the function “/processValue” at the end. According to the data model of ia:, further functions are described in chapter 6.2 as possibilities for other future developments.

Results

4.3 Implementation of the visualization

The dashboards of the two software stacks look almost identical to the user. The only differences are in the implementation of the dashboard. This chapter introduces the main dashboard and its various areas. Besides, the differences in the implementation of the two software stacks should be shown. The data displayed always relates to the period selected in the dashboard.

The main dashboard has various areas that provide the user with a wide range of information. Figure 43 shows the main overview of activities and expiring states. The entire signal is shown in the upper area (1), which contains all occurring tag values. This area is intended to provide the user with a better understanding. Also, the number of measured values per sensor is displayed for the viewed time window. In addition, the number of measured values is displayed according to the two available downsampling levels. In the upper edge, there is also a graph showing the vibration values of the sensor installed at the workstation of the LEAD Factory. In a further sub-area (2), information over two discrete panels is displayed. The upper panel shows all activities carried out on the workstation with colored lines and thus offers real-time tracking. The panel is also based on the overall signal with the tag values. The second panel shows the six different states of the workflow and the associated breaks. The duration of the states can already be taken from the corresponding length of the bars. In the third sub-area (3), the type of the last produced scooter is displayed. Besides, the history of the produced scooter types can be taken from a discrete panel. To determine this, either the removal of the different colored wheels or the RFID product information can be used.

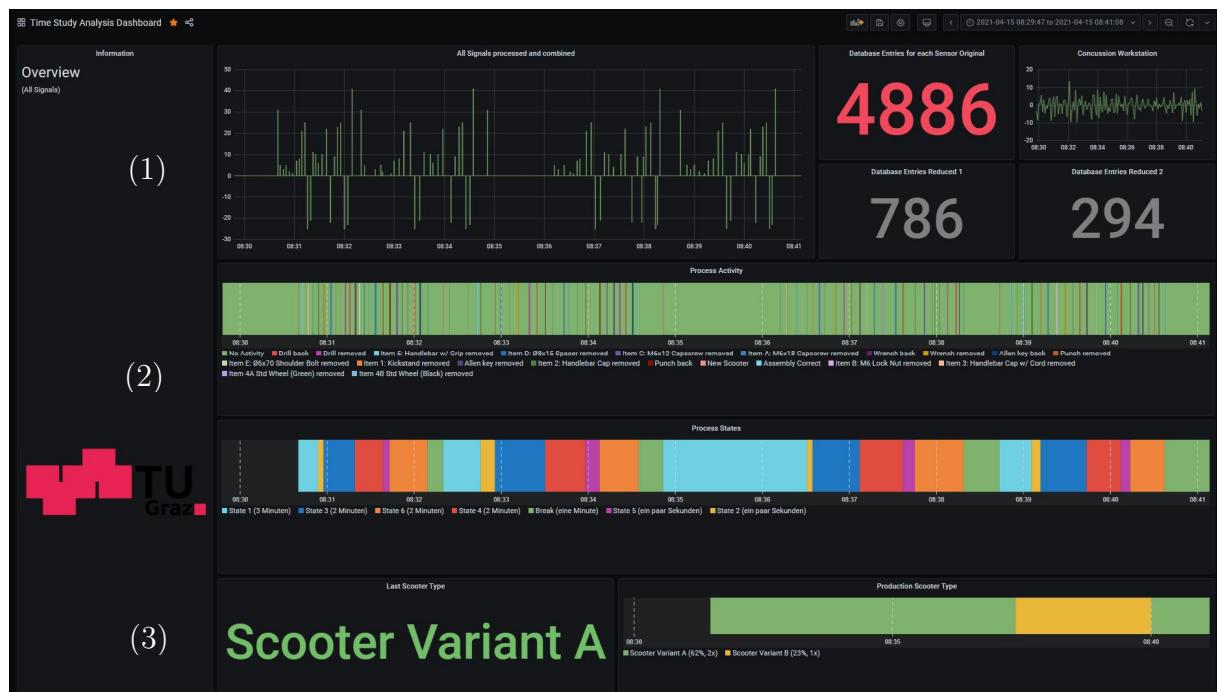


Figure 43. Main overview of activity tracking on the dashboard

Results

Various graphs, like shown in Figure 44, are used to visualize the multiple time measurements and to be able to draw conclusions over extensive periods. In the upper area (4), the total assembly time of a scooter at the workstation and the break between the end and the beginning of a new scooter assembly are shown in two graphs. Besides, the average assembly time, the last assembly time, and the number of installed scooters are displayed. The individual time values of the assembly can also be viewed in a table. The user can choose between assembly and break times. In section (5), the times measured for the six states are plotted in a table with the associated timestamp and marked in color in the event of negative and positive exceedances. To better visualize these measured values, they are shown in single graphs for each state under section (6). To make better statements about the development of the measured values over a more extended period, the measured values are smoothed at this point by an exponential moving average. This EMA makes the trend component visible. On the other hand, possible extreme values can be seen in the previous table using the colored marking. In addition to the graphical representation, the average times and the last time measurement of the states are shown.

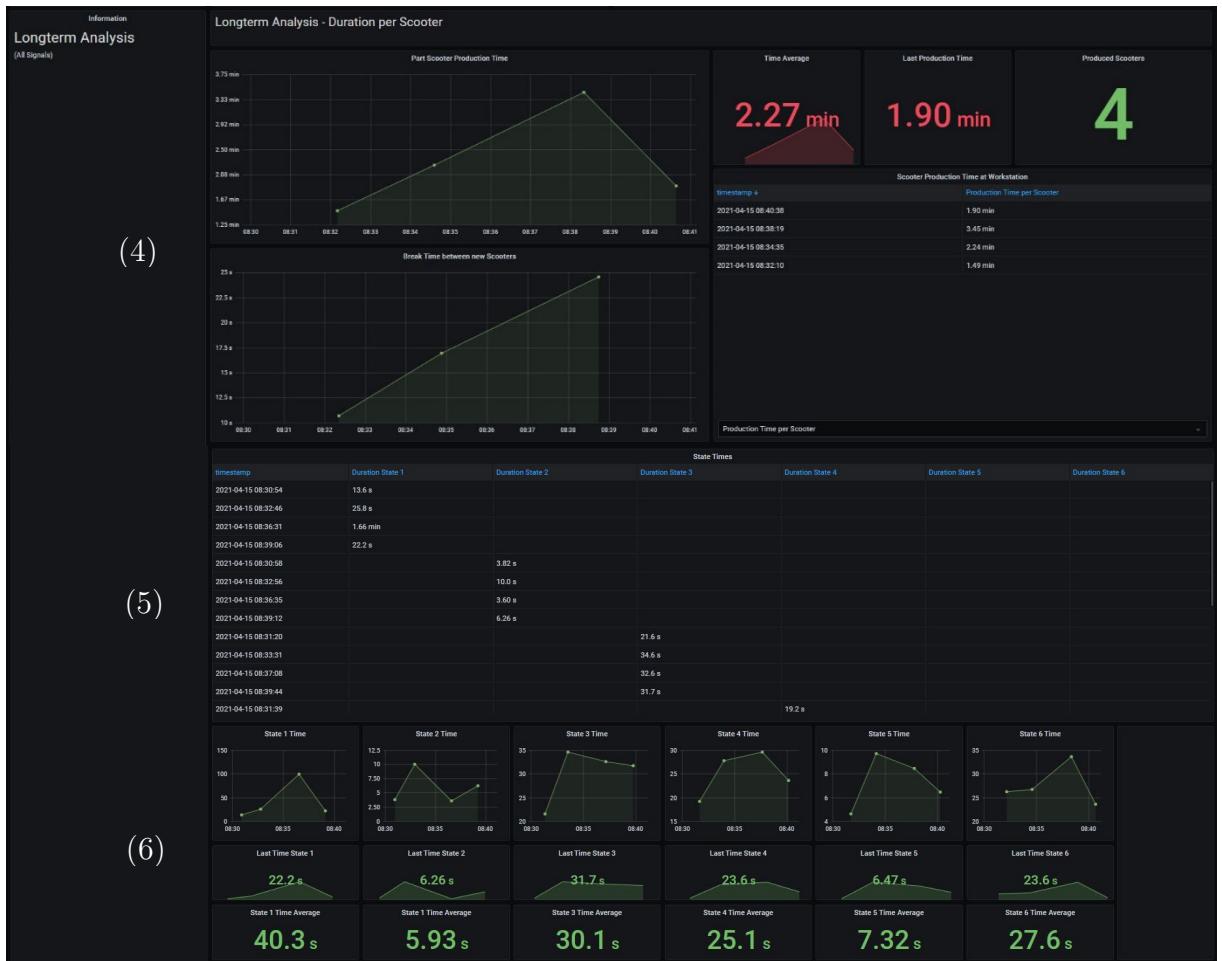


Figure 44. Longterm scooter production data

Figure 45 shows a dashboard section (7) for analyzing defective parts during assembly. For this purpose, a histogram was selected for better visualization, fed from 11 different

Results

data sources. The histogram shows the frequency and distribution of specific defective materials at the workstation. Besides, the total number of faulty components and the component last declared as defective are displayed. All parts registered as defective are listed with the associated timestamp for further tracking in an additional table.

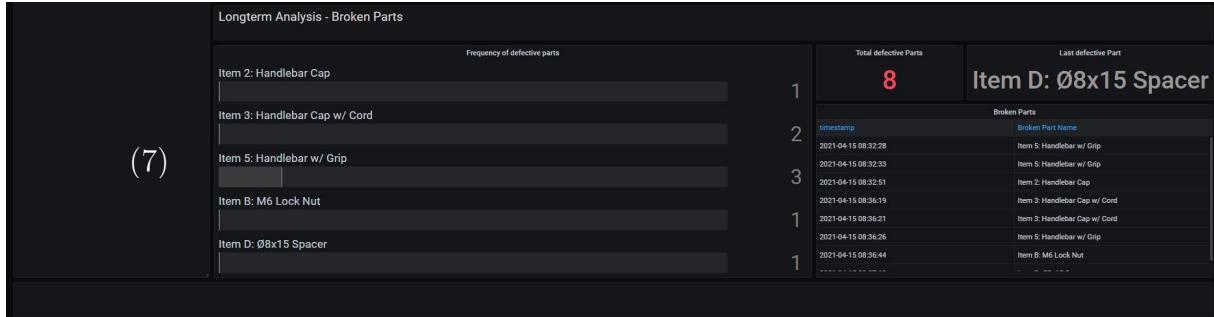


Figure 45. Display of defective components and their distribution

In a further section (8), as shown in Figure 45, various additional information is directed to the user with discrete panels. These are the different sensors connected to the system. The figure shows the activities of the position sensors and the induction sensor as an example. Other sensors are also listed. In addition to the main dashboard described here, other dashboards are available for the user, such as a home dashboard or dashboards for visualizing the data processing steps of the individual sensors.

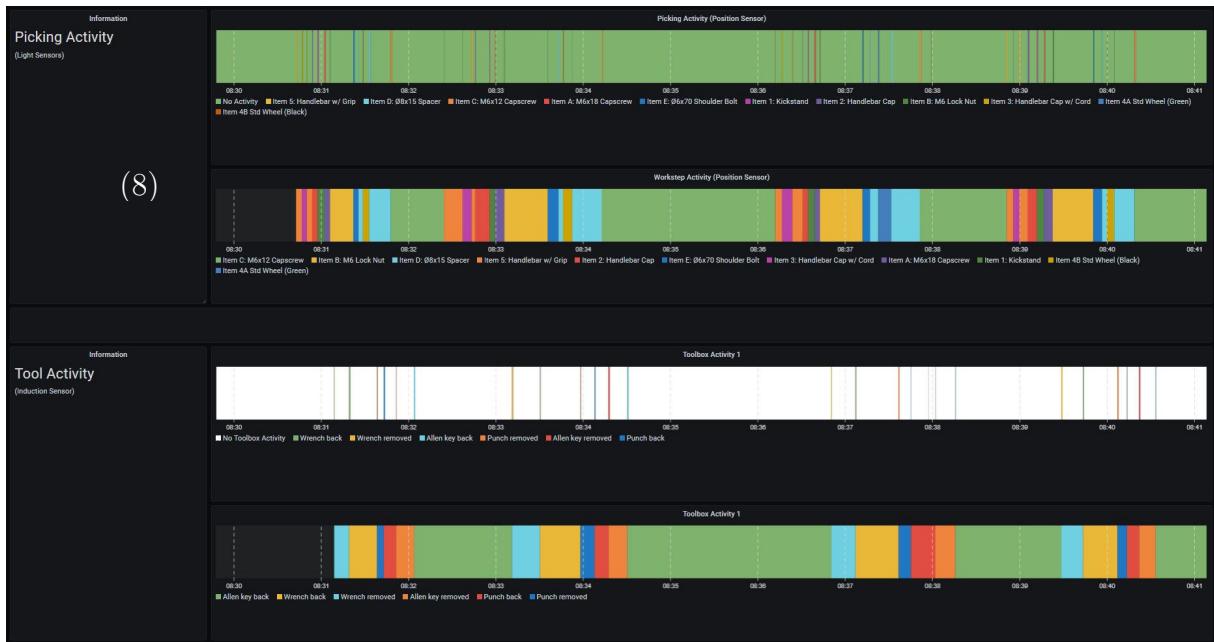


Figure 46. Display of the individual sensor activities

The dashboards appear identical to the user of the two software stacks, but the implementation of the dashboards is different. This begins with the transfer of the required data to Grafana. While with Influx, each interface, as shown in Figure 47 (left), has to be defined and configured separately, in software stack 2 in Grafana, it can be chosen directly from the respective topics in drop-down menus. The interface configuration includes the specification of the data source type, a URL, the bucket, and

Results

a token. This token must first be created in Influx. The interfaces implemented between the systems and Grafana can be found in Chapter 3.3.

The figure consists of two screenshots. The left screenshot shows the 'Data Sources / InfluxDB Datenbank Combined' configuration in Grafana. It includes sections for 'HTTP' (URL: http://influxdbV2), 'Auth' (Basic auth, TLS Client Auth, Skip TLS Verify, Forward OAuth Identity), and 'InfluxDB 2.0.0 Details' (Organization: ia, Default Bucket: metrics_join_1, Token: configured). The right screenshot shows a Flux query editor in Grafana. The query is:

```
from(bucket: "Test2")
|> range($range)
|> limit(n:1000)
|> filter(fn: (r) =>
    (r._measurement == "complete"))
|> filter(fn: (r) =>
    (r._field == "Duration"))
|> elapsed(unit: 1s)
|> map(fn: (r) =>
    ({r with _value: float(v: r.elapsed)}))
|> filter(fn: (r) =>
    (r._value >= 4) and (r._value <= 100))
|> drop(columns: ["elapsed"])
|> exponentialMovingAverage(n: 3)
```

The 'FORMAT AS' dropdown is set to 'Result tables'.

Figure 47. Definition of a Grafana interface (left) and Implementation of a graph data source (right) with Influx

In combination with InfluxDB, the query language flux must be used in Grafana through a plug-in. Figure 47 (right) shows an exemplary Flux query for a graph from section (6). The use of Flux opens up further possibilities for changing the data, but it means that only people with in-depth knowledge of Flux can create graphs. Besides, the creation of queries is significantly more time-consuming. With software stack 2, this active query option is no longer available. In Grafana, only fundamental transformations of the data can be carried out, such as filtering data, value mapping, or renaming columns and rows. Grafana also offers the user the option of performing essential mathematical functions. All these settings are made in this Grafana version in simple dropdown menus. Overall, this leads to easier handling but limits the possibilities for further data processing with Grafana. Almost all data processing must therefore already be carried out and completed in node-red.

4.4 Comparison of the two solutions with different software-stacks

After presenting both system implementations, these are to be compared based on selected criteria as part of a utility value analysis. Finally, based on this, a well-founded recommendation for one of the two systems should be made. The procedure for creating this utility analysis is based on the guidelines of Prof. Dr. Jörg B. Kühnapfel, which he describes in his book “Utility analyzes in marketing and sales” [73].

Results

Definition of the criteria and their evaluation

Criteria for evaluating the two systems were collected and discussed in a brainstorming session. In the end, ten different criteria could be memorized, as shown in Table 6.

Table 6. Presentation of the evaluated comparison criteria

| Criteria | Description |
|--|---|
| 1. Usability | The system is easy to use and therefore extremely user-friendly |
| 2. High failure safety | The system is highly fail-safe and therefore has a long operating time |
| 3. Low implementation effort | Low effort for the user in terms of code programming |
| 4. Low complexity of the data processing | A low level of complexity characterizes the process of data processing in the system |
| 5. System expandability | The system can be expanded as required |
| 6. High data processing speed | The system enables a high speed in the data processing |
| 7. High number and variety of possible programming languages | The large number and variety of possible interfaces to other systems for the exchange of data |
| 8. Good support and community | Good support from official bodies and a large support community to help with implementation, troubleshooting, templates, and much more. |
| 9. Used Data model | The data model used by the system (e.g., relational, NoSQL) |
| 10. Used Query Language | The used query language of the database system |

The just described criteria must be weighted to implement a utility value analysis concerning their importance for the decision. This results in relative importance for each criterion. The weighting of the criteria is given in percent. In total, 100% results for all weights of the criteria. The pair comparison method is used to enable the most objective possible weighting. Therefore, the entire weighting is fragmented into small individual decisions. For this purpose, the criteria are plotted in a matrix or cross table, as shown in Table 7. All criteria can then be compared in pairs, and the following values can be assigned:

0: The row value is less important.

1: The row value is more important.

The table reads as follows: “Usability is less important than Failure safety” or “System expandability is equally important than Complexity of data processing.” The left column is always mentioned first.

Results

Table 7. Matrix for pair comparison method completed by the author

| | Usability | Failure safety | Implementation effort | Complexity of data processing | System expandability | Data processing speed | Possible interfaces | Support and community | Data model | Query Language | Total Points | Weight [%] |
|-------------------------------|-----------|----------------|-----------------------|-------------------------------|----------------------|-----------------------|---------------------|-----------------------|------------|----------------|--------------|------------|
| Usability | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 4 | 8.89 |
| Failure safety | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9 | 20.0 |
| Implementation effort | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 4.44 |
| Complexity of data processing | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 2 | 4.44 |
| System expandability | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 7 | 15.56 |
| Data processing speed | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 | 8.89 |
| Programming language | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 5 | 11.11 |
| Support and community | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 8 | 17.78 |
| Data model | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2.22 |
| Query Language | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 6.67 |

After all pair comparisons have been made, each criterion's sum of points is calculated. This sum is also shown in the table. The weighting of the criteria can be calculated from the point values using the rule of three. To obtain an even more objective weighting, the paired comparison was carried out by a total of 11 participants with sufficient technical knowledge. A more detailed breakdown of the individual results can be found in Appendix D. The determined weightings were offset to an average weighting, which is shown in Table 8.

Table 8. Weighting of the criteria after interviewing 11 participants

| Criteria | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|------------|-------|-------|------|------|-------|------|-------|-------|------|------|
| Weight [%] | 11.11 | 18.18 | 6.06 | 6.46 | 13.13 | 8.89 | 12.93 | 12.53 | 5.66 | 5.05 |
| Rank | 5 | 1 | 8 | 7 | 2 | 6 | 3 | 4 | 9 | 10 |

Looking more closely at the resulting weighting, it is particularly noticeable that the maximum differs significantly from the remaining weightings concerning the distances. The highest weighting is assigned to the criteria "High failure safety". During the entire implementation of the project, it became clear that this criterion is of enormous importance in practice. A system that does not run consistently and safely may not be suitable for use and therefore almost useless. All participants agreed on this. There were isolated outliers up and down in the weighting for most other criteria. In conclusion, it can be said that the criteria of usability, high failure safety, system expandability, the

Results

high number of possible interfaces as well as good support and community are viewed as above-average relevant. On the other hand, the remaining five criteria are below average. It is only noticed that the four criteria with the lowest relevance have an almost identical weighting.

Evaluation of the decision criteria and utility value calculation

In the next step, the two solution variants are evaluated in terms of the described criteria using a school grade scale from 1 to 6, as in Table 9. In doing so, however, the school grades must be assigned a reverse value. One represents the lowest and six the highest evaluation. The evaluation is entered in the respective evaluation column. A rating is then determined using the weighting and the previously determined evaluation by multiplication. This is again entered in the associated column. The sum of the rating columns results in a final score for the respective software stack.

Usability: The usability criterion is difficult or impossible to quantify and is therefore subject to a certain degree of subjective influence. Quantifiability would only be possible with great effort through surveys or tests. Due to the private beta phase, no users can be questioned. To be able to carry out the most objective assessment despite these circumstances, short user diaries were kept in the course of the project. Positive and negative events in work with the software stacks were noted. In the aforementioned school grading system, grade 6 can be assigned the property “system can be used in a self-explanatory manner,” and grade 1 can be assigned the property “system cannot be used by users.” Comparing the two systems step by step concerning this criterion, it can be seen that the usability of software stack 2 must be viewed as significantly higher. The factorycube can be switched on and off with stack 2 without any problems. With stack 1, the systems have to be re-initialized. Programming in the influx database requires a high level of knowledge, as settings have to be made in many sub-categories, and many components interact with one another. This is solved graphically in node-red, blocks can be easily integrated by drag and drop. While source code must be used in Grafana with stack 1, with stack 2, it can be chosen from dropdown menus.

High failure safety: The weighting of the criteria has shown that high importance is attached to the high failure safety criterion. As already known, software stack 1 had to be changed to stack 2 due to the complete failure. Considering the criterion, the severity of security incidents and the system's availability must be used for the assessment. In stack 2, many applications such as Kubernetes, K3os Rancher, Lens, and MQTT Explorer have significantly improved system security. With Kubernetes and K3os Rancher, the individual containers can be operated better. With Lens, the containers can be checked for their function and the correct data flow with an MQTT Explorer. Due to the total failure of the system with stack 1, this criterion can only be rated as “unsatisfactory” (1). With stack 2, the evaluation can be carried out by calculating the availability. The availability is calculated according to:

Results

$\text{Availability}(\%) = (1 - (\text{downtime} / (\text{production time} + \text{downtime}))) * 100\%$. With a downtime of 2 days and a production time of 16 days, the current availability is 87.5%. If this percentage is converted into school grades, this results in a “good” (5) rating. This is only a snapshot. In the long term, the value of the availability will increase significantly, as bugs are continuously being fixed in the current beta phase, which has had a negative impact on availability until now.

Low implementation effort: This criterion is also difficult to quantify. One possibility is counting lines of code (LoC), whereby such counting is not possible due to the graphical programming in node-red. A comparison with similar programming solutions is only possible to a limited extent due to the exceptional software setup. At this point, the aforementioned user diary can be used to determine the required time for programming to assess based on this. The essential requirement for comparability is that both systems were initially familiar or unknown to the programmer. The programming in influx via tasks and source code clearly shows that the implementation effort in node-red is significantly lower due to the graphical programming. Besides, community templates can be used in node-red to reduce the effort further. Therefore, a low programming effort concerning influx is only possible with great difficulty. In contrast to node-red and comparable projects, the effort with influx can be regarded as average.

Low complexity of the data processing: To evaluate the complexity of software products, comparison projects should be considered. The two implementations in the two software stacks themselves fulfill this purpose. Besides, the effort and the readability of the implementation can be used again. This raises the question of the comprehensibility of the overall system. In the context of this chapter and the creation of the architecture under 3.3, it became clear that data processing via influx is much more complex. Due to the interlinking of individual tasks and the repeated intermediate storage of data, it is much more challenging to familiarize yourself with such a system. In node-red, on the other hand, the data flow and thus the data processing can be easily traced. An essential difference arises in the complexity of the further data processing steps, which could be implemented in node-red in a much simpler and more structured way. Furthermore, there is no need to set up and manage interfaces. The evaluation is based on the most uncomplicated possible design and complexity of such a time recording system.

System expandability: Concerning the expansion of the system, the possible enlargement of the database and the possibility of new types of data are essential criteria. Both Influx and Timescale can be expanded almost at will because of their architecture. The only limiting factor is the physically available storage space of the factorycube, which can be expanded without any problems. Thanks to its non-relational data model, InfluxDB also supports the storage of non-tabular data such as graphics or documents. TimescaleDB does not support this storage. This disadvantage will be eliminated in the future by separate programs of the factorycube, such as programs for processing camera images. The possibilities of system expandability can therefore be regarded as equivalent.

Results

High data processing speed: To handle a large amount of data in real-time, a high data processing speed is desired. Influx represents the slightly faster database in the setup operated here in terms of both, the ingest and query rates. Overall, the data processing in influx is slower due to the necessary caching. For this purpose, tests were carried out on the setup. However, a specific statement cannot be made, as the speed depends mainly on framework conditions such as the current amount of data or the system load. Even the comparison with other database systems does not allow any quantifiable statements. For this purpose, all databases to be examined would have to be tested and compared in the same setup, i.e., same CPU, memory size, data volume, type of data, and much more. In literature, only rough estimates can be found, according to the processing speed, in which both Timescale and Influx are in the middle range. Both systems are equally suitable for the tasks performed here.

High number and variety of possible programming languages: To be able to read out as many data sources as possible and to be able to guarantee a high degree of variability, the greatest possible number of supported programming languages is necessary. Comparing the 20 most common time-series databases, the result ranges from one to 18 supported programming languages. Influx supports 16 and Timescale 13 possible programming languages. If you categorize these values in the range above, the resulting rating for Influx is “very good” and “good” for Timescale. Nevertheless, it must be mentioned at this point that this assessment basically concerns the database systems. If necessary, the options can be expanded through self-programmed interfaces and add-ons.

Good support and community: A large and active community, in addition to the manufacturer’s support for programmers, is very important for familiarization, troubleshooting, and especially about suggested solutions and examples. The first software stack was a closed software project, so little information about the hardware and software was available. Furthermore, due to the new introduction, only documentation from the developer could be used for the associated database language. In the event of unsolvable problems, the manufacturer’s support had to be accessed immediately. This made the practical work much more difficult. Software stack 2, on the other hand, is designed as an open-source project. The entire software is available on GitHub and can be viewed and changed. Besides, ia: offers comprehensive documentation on how to use the software. With any problems, the community on GitHub can now be consulted. If problems occur frequently, these may have already been described by other users on the platform. This speeds up practical work with the system enormously. Furthermore, node-red offers comprehensive documentation and a GitHub forum for users. However, since setting up a community is time-consuming and the software is still in the private beta phase, only a “satisfactory” rating can be given here.

Results

Used Data model: The two databases used here are two opposing data models. At this point, no assessment should take place as to which data model is fundamentally better or worse. Such an assessment would simply not be possible but can only be clarified for a specific application. This raises the question of which data model is more suitable for this type of application and its data. In principle, both databases and thus data models are suitable for this application, as both were specially developed for time series data. There is a slight advantage for the relational data model used by Timescale due to the consumption of significantly lower storage capacity (factor approx. 50%). Due to their suitability, both models can be assessed as “good” or “very good” for this purpose.

Used Query Language: Concerning this criterion, on the one hand, the query language Flux and the graphical programming via node-red and javascript must be used. Timescale works with SQL, but this query language is only used very rarely in the context of this work and can therefore be neglected. Consequently, it is important to clarify which of the two programming languages is more suitable for this project. Many of the advantages of programming via node-red have already been explained in the previous chapter. Graphical programming enables even inexperienced programmers to work productively in a short time. On the other hand, Flux requires a significantly more extended training period. Furthermore, node-red offers an extension with javascript, which is well known. Information on javascript is easy to find. In terms of legibility, both programming languages can be seen as equal, as special attention was paid to this during the development of Flux. Both programming languages can quickly be used productively after training, whereby node-red offers the larger range of functions, and its graphical programming surface makes operation simple and intuitive. An objective assessment is challenging to make, especially with this criterion.

Table 9. Utility analysis for the two factorycube software stacks

| Criteria | Weight | Software stack 1 (Influx) | | Software stack 2 (Timescale) | |
|-------------------------------|-------------|------------------------------|--------------|---------------------------------|--------------|
| | | Evaluation | Rating | Evaluation | Rating |
| Usability | 11.11% | 3 | 0.333 | 5 | 0.556 |
| Failure safety | 18.18% | 1 | 0.182 | 5 | 0.909 |
| Implementation effort | 6.06% | 4 | 0.242 | 6 | 0.364 |
| Complexity of data processing | 6.46% | 3 | 0.194 | 5 | 0.323 |
| System expandability | 13.13% | 4 | 0.525 | 4 | 0.525 |
| Data processing speed | 8.89% | 4 | 0.356 | 3 | 0.267 |
| Programming languages | 12.93% | 6 | 0.776 | 5 | 0.647 |
| Support and community | 12.53% | 1 | 0.125 | 4 | 0.501 |
| Data model | 5.66% | 5 | 0.283 | 6 | 0.340 |
| Query Language | 5.05% | 4 | 0.202 | 6 | 0.303 |
| Total / Score | 100% | | 3.218 | | 4.733 |

Results

Finally, Table 9 shows the final rating of the two software stacks in the bottom line. Stack 1 achieves a score of 3.218, and stack 2 a score of 4.733. In terms of grades, stack 1 reaches a sufficient result and stack 2 a good result. This rating clearly shows that the switch to software stack 2 was the right decision. A significantly better result was achieved, especially in the critical categories of “high failure safety” and “support and community”. Nevertheless, it is noticeable that a large number of positive grades were given. This fact leads to the conclusion that the developers at ia: also used similar or identical criteria for selecting the various software components. Finally, it should be noted again that a certain subjectivity cannot be avoided due to the criteria which are difficult or impossible to quantify.

4.5 Conception of training and teaching opportunities

In addition to implementing a digital time recording, the factorycube is to be integrated into teaching within the framework of the learning factory. The use of the factorycube combines the subject areas “sensor technology”, “dashboards”, and “data analysis and interpretation”, which are essential areas in training to face the challenges of digitization. As already described in Chapter 2, a distinction is made between three different levels in the conception of such training courses, the macro, meso, and micro levels. The learning environment, the macro level, forms the learning factory. At the meso level, the individual training modules are defined, and a competency transformation is carried out. Table 10 shows the development of training modules with the factorycube. In the beginning, the primary competence is defined, which each participant should learn through the training. This competence can, in turn, be divided into several subcompetences, which contain corresponding actions on the part of the participants. The participants need a knowledge base tailored to this to carry out the actions. The meso level described in Table 10 illustrates a training course in which the participants recognize the potential for data generation with sensors in the production environment and then select and install the required sensors. The elementary data processing with node-red and the visualization with Grafana should also be brought closer to the participants. The actions described in the table are divided into small actions to create learning situations in the micro-level to be carried out. These actions are integrated into an action setting and form different exercises for the participants. In the future, this training should take place in the LEAD Factory of the IIM and will be tested by students and company members of different industry sectors.

Results

Table 10. Didactical transformation of sensor training with the factorycube

| Competence | Subcompetences | Actions | Knowledge Base |
|--|---|---|--|
| Participants have the ability to gather, process and visualize sensor data from the production process | Participants are able to identify possible use cases for sensor implementation in the production process | *Analysis of the production system *Plan and discuss which data would be useful to derive *Plan and discuss how and where such data can be derived | *Possible potentials of data sources *Use cases of data sources (best practice) *Framework or evaluation sheet for choosing suitable use cases |
| | Participants are able to choose suitable sensors for the use cases | *Evaluation of the available sensors *Selection of a suitable sensor *Positioning and mounting of the sensor *Correct wiring of the sensor | *Framework or evaluation sheet for choosing the suitable sensor *Sensors and their functionalities on the market *Setup of sensors |
| | Participants are able to understand the data flow from the detection of sensor values to their visualization | *Experience the data flow process | *Know basic elements of the data flow process (rough overview) *Know functionalities of the single data flow elements |
| | Participants are able to generate sensor data in the production system and read in the data via a flow editor | *Identify and control interfaces via an MQTT explorer *Define and implement interfaces in the flow editor (input) | *Knowledge of interfaces and communication protocols *Basic elements of the flow editor and their functionalities |
| | Participants are able to transform and process sensor data derived from the production system | *Transform input data to a suitable data type *Implement an aggregate function with suitable settings *Construct and implement a logic to form a difference | *Data types (object, string, number, array) *Basic JavaScript functions *Downsampling methods (aggregate window) *Filtering methods |
| | Participants are able to transfer processed data via a communication protocol | *Transformation of outgoing data in the right format *Define and implement interfaces in the flow editor (output) *Control interfaces via an MQTT explorer | *Communication protocols and their functionalities *Data model of the transferred data |

Results

| | | | |
|--|---|---|---|
| | <p>Participants are able to visualize and understand processed sensor data via dashboards</p> | <ul style="list-style-type: none">*Selection of a suitable panel type*Create and develop dashboard panels*Visualize and understand data*Adding and visualization of thresholds | <ul style="list-style-type: none">*Basics of visualization with dashboards and panels*Function of dashboards and panels*Creation and development of a dashboard |
|--|---|---|---|

4.6 Summary

In this chapter, the previously defined architectures of a digital time measurement with two different software stacks were implemented with the help of the factorycube, and the approach and structure were presented. The required process steps for the individual sensors to collect and process data were explained and demonstrated. The design and function of the dashboard for evaluating the results were also described. The main focus was on the differences between the two used systems. It turned out that software stack 2 has significant advantages in terms of various criteria such as system security, usability, or the involvement of community and support. A survey enabled these criteria to be weighed against each other. It served as the basis for a utility value analysis, which confirmed the previously subjective impression as objectively as possible. Therefore, the use of software stack 2 is the more suitable variant. Finally, a basic training course for teaching content in the field of sensor technology was developed and created, which will be used in teaching within the framework of the IIM's LEAD Factory.

5 Discussion

In this chapter, the research questions defined at the beginning should be answered as best as possible based on the knowledge gained in chapters 3 and 4. Besides, the existing limitations that occurred during the work are considered, which led to the fundamental change in the software system. This chapter cannot answer in detail all problems, ideas, improvements, and questions that have arisen; rather, it is intended to concentrate on the main research questions.

5.1 Limitations and related software update

At the start of the project, the very new and error-prone software stack 1 was used. There were repeated problems, some of which made it impossible to operate the system. Each time the system was restarted, an SSH connection had to be established, and the software initialized. Therefore, the program container had to be started and the current system time assigned to the system. Carrying out the initialization is not overly complex but requires the necessary knowledge. Besides, when using software stack 1, there were very limited data processing options with the InfluxDB database. The main reason here is that programming variables cannot be used.

During the project, it was necessary to switch to the new software stack 2 because software problems arose which could not be resolved even by ia:. The security certificates of IP addresses were not recognized. This meant that no more data could be transferred between the IO-Link system and the database. The function of the system was no longer given. Due to the limitation and the problems that occurred, it was therefore decided to switch to a new software system. An update was carried out together with Industrial Analytics. Software stack 2 is a not yet published software package that the IIM could test as first participants in a private beta phase. The software tools such as Kubernetes and Lens described in the previous chapter have largely eliminated the system's limitations. The exchange of constant feedback to the ia: development team was because of the private beta expressly requested. This relates mainly to possible system bugs, practical work with the system, the use of the newly created documentation, and possible suggestions for improvement. Bugs and related solutions were shared on the network-based platform for version management for software development projects GitHub in a private repository, as shown in Figure 48.

After switching to the private beta software version, the changed structure with the transfer of the processed data to the ia: server resulted in another problem. An internet connection is required for this process. This requirement initially made it impossible to integrate the data from the LEAD Factory. Since the wifi of the LEAD-Factory has no internet access, the factorycube would have to be connected to two networks

Discussion

simultaneously to be able to access the data from the learning factory on the one hand and to transmit it to the server on the other. The router of the factorycube does not support such multiple connections. As a solution, a local server stack could be implemented with the help of industrial analytics. This server stack saves the processed data on a local server, which means no permanent internet connection is required. The factorycube can therefore be connected to the LEAD Factory network in order to be able to access the required data. All other functions can be carried out at the same time. The Grafana dashboard can also be accessed locally via an IP address and can therefore be called up and displayed on all end devices of the learning factory.

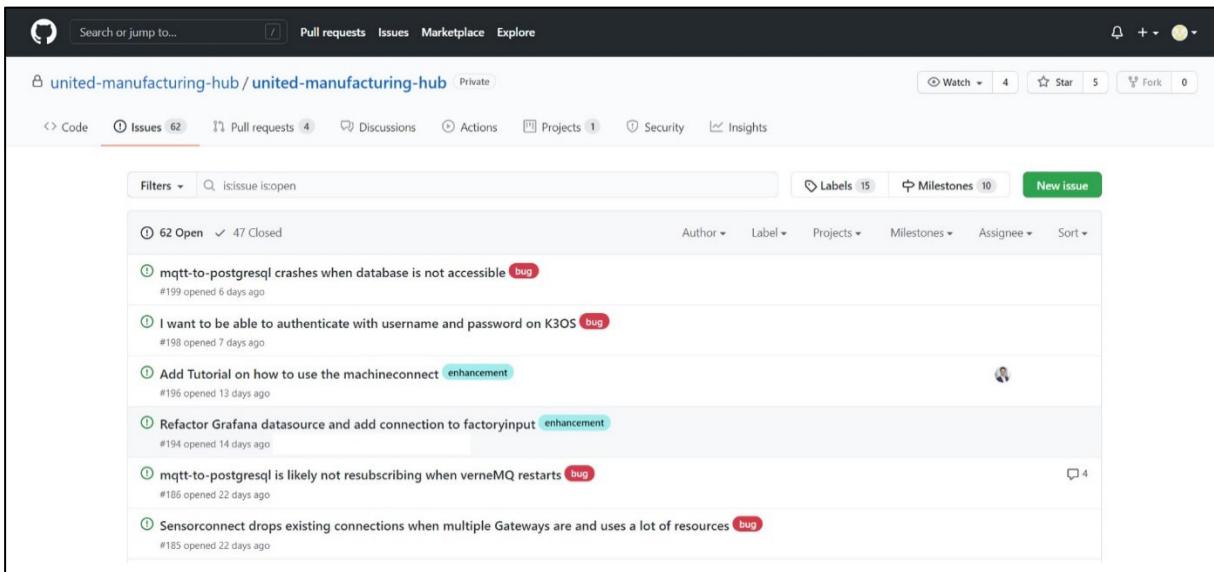


Figure 48. Collaboration with ia: as part of the private beta via GitHub

5.2 Answering the research questions

In the following, the research questions defined at the beginning of the thesis are taken up, discussed, and answered as far as possible based on previous chapters.

5.1.1 Research question 1

As the first research question was defined:

“Which software stack, including data model (relational or NoSQL) for storing data in time-series databases, is most suitable for performing a time study analysis for manual activities, and how is this reflected in the practical implementation using the two different databases TimescaleDB and InfluxDB?”

The first part of the research question can be answered very clearly. Software stack 2 with the significant components TimescaleDB and node-red is the much more suitable setup for digital time recording. This statement can be made by performing a utility

Discussion

analysis in Chapter 4. The two stacks were rated based on ten different criteria, which were previously weighted among each other through a survey under eleven participants. Only in the categories high data processing speed and number of supported programming languages, stack 1 could achieve a better result than stack 2. All other criteria speak in favor of stack 2. In particular, with regard to the two above-average weighted criteria, high failure safety and support and community, stack 2 achieved a significantly better rating. At 18.18%, high failure safety was the most heavily weighted criterion, which deviates significantly from one another with ratings of one and five. As already explained in chapter 5.1, the poor performance of stack 1 concerning this criterion was the decisive point for switching the system to software stack 2. All the improvements made to the system regarding the ten defined criteria ultimately result in a score of 4.733 for stack 2. Compared to stack 1 with just 3.218 points, this is a significantly better result. However, answering the second part of the research question is much more difficult. It isn't easy to make a clear assessment in favor of a data model. The evaluation of the data model was discussed, and the difficulties involved were described while the utility analysis was being carried out. Due to the basically tabular form of the IoT sensor data, the relational data model of TimescaleDB does not reach its limits, especially since the intermediate storage of data is not necessary with this concept. Simultaneously, the potential of InfluxDB's NoSQL approach is rarely used. Therefore, both systems are equally suitable for the application of digital time recording. However, if the format of the data changes in the future, i.e., graphics, images, or videos, are to be saved, this can only be implemented with a non-relational approach. The main difference between the two DBMS lies in the area of data processing and software architecture. Due to the use of tasks with intermediate storage of the data, the processing of the data with stack 1 is much more complex and networked compared to the implementation of the processing with node-red. This fact flows into the criteria of usability, low implementation effort, and complexity of data processing. Answering this question leads us to research question number 2.

5.1.2 Research question 2

After answering the question of the better system, research question 2 should now answer questions about the implementation of digital time recording, which would be:

“Can existing data from the LEAD Factory be used, and which methods in data processing must be developed to perform a time study analysis with time-series data?”

After eliminating countless problems, the first part of this research question can be answered with yes. This answer applies to both software stack 1 and stack 2. This question can be seen and answered from two different perspectives. In both stacks, it is technically possible to transfer the data from the LEAD Factory into the respective DBMS of the two stacks. From the point of view of digital time recording, it is also

possible to use the data from the LEAD Factory in the process of data processing as an information source. The LEAD Factory data is stored on the learning factory's server in an MSSQL database and is transferred from there to the factorycube. In stack 1, this is solved with an add-in of a task, in stack 2 with a special node in the graphical programming software node-red. In the case of stack 2, the connection with the network of the learning factory turned out to be complex, as an internet connection was necessary. This problem could be remedied by implementing a local software stack. The second part of the research question deals with processing the data itself and its methods. The procedure for processing and evaluating the sensor data, in particular, was combined and networked with several existing methods. The mathematical methods most frequently used were moving averages, aggregate windows, and differences. Besides, a two-step procedure for eliminating values that were viewed as incorrect was developed and implemented. This procedure makes it possible to obtain unambiguous and thus interpretable sensor values. Methods also had to be developed for the further processing of these values. These methods differ significantly from one another depending on the used stack. In stack 1, for example, a method for detecting the duration of work could be implemented by assigning individual values to several states. In stack 2, this could be realized through the use of arrays and the combination of tag values with measured times. This work confirmed that there is only very limited information on this topic in the literature, which leads to the need for research question number 3.

5.1.3 Research question 3

After answering research questions on the practical implementation of a digital time recording, a further question regarding training opportunities in the vicinity of a learning factory should now be answered.

“Can the training of participants by using a factorycube represent a suitable way for imparting knowledge regarding digitization in the field of IoT time-series data analytics?”

This question can also be answered with yes. As part of this work, a basic training in the field of sensor technology could be designed, which is based on the platform of the factorycube and its sensors. As a setup for this, the IIM's LEAD Factory offers a suitable environment. The participants get an insight into aspects of the topic of sensor technology from the determination of digitization potential, the choice and assembly of appropriate sensors, the transmission and processing of data up to their visualization using dashboards. This insight provides a holistic overview of the subject area of sensor technology and can thus serve as an introduction to the subject area for the participants. A validation of the training concept in practice is still pending due to the restrictions of the Covid-19 pandemic. Subsequently, further statements and possible improvements to the training concept can be made.

6 Conclusion

In this thesis, various aspects of digitized time measurement with time series-based data were discussed. Thus, the work can serve the reader as a basis for getting started with this topic. After answering the research questions, the work is to be summarized again in this chapter, and the most important findings are listed. Besides, further thoughts for future improvements or work should be presented to make this topic even more holistic.

6.1 Summary

At the beginning of the work, the motivation for carrying out a digital time measurement in the vicinity of a mechanical production facility and the learning factory of the IIM at Graz University of Technology was presented. Based on this, three main research questions could be defined as the aim of this work.

In a literature research on the main topics of this work in Chapter 2, a research gap was confirmed concerning the defined research questions. In addition, important information was provided for the reader to understand this work. This includes the definition of a learning factory and the development of training opportunities in its environment. The fields of IoT, Big Data, and RFID could be identified as other essential topics. In the representation of database systems, the different types of systems were explained, and their differences were shown. There are relational and non-relational approaches, which are mainly called NoSQL databases. TSDBs have been declared as a special type of DBMS. Two TSDBs from different developers and their query languages were presented in the following. The basics of data visualization, as well as selected basics of data analytics, could be examined more closely. In particular, the connection between data analytics and time-series data was made clear by the component model. Finally, the structure and process of a time recording could be shown using a schematic flow diagram.

At the beginning of Chapter 3, the experimental setup for performing the automated time recording was presented. This setup consists of the factorycube from the start-up ia:, various IO-Link sensors and existing sensors from the LEAD Factory. In addition to the various hardware components of the factorycube, the individual software components were also presented. In the case of the existing sensors at the LEAD Factory, the RFID system, in particular, was examined, and its function was explained. A suitable workstation of the learning factory was selected to prepare the time recording, and the associated work steps were discussed. Based on the defined work steps, suitable sensors could be chosen for their detection and integrated into the workstation. The entire workflow was divided into defined states for time recording, and various tag values were assigned to the individual activities. Thus, the same basis could be created to execute the time measurement with two different software stacks. As a result, the respective

Conclusion

software architectures were subsequently designed. Due to the complex structure, a BPMN was prepared for software stack 1, which visualizes the interaction of the various program tasks and storage locations. The architecture for software stack 2 could be represented in a simplified way by the graphical programming surface of node-red. The significantly leaner architecture of stack 2 was already evident at this point.

In chapter 4, the practical implementation of the time recording with the two different software stacks was shown. The data collection via a telegraf agent and a flux task for stack 1 were explained in the beginning. The 15 different tasks implemented with the Influx database could be presented. Besides, the function of these could be described using flow charts. The data collection in connection with stack 2, the data could be transferred directly to node-red via MQTT. The data processing logic was based on the procedure in stack 1, so the differences and extensions of the data processing could be shown at this point. Afterward, the data could be visualized with a dashboard in Grafana, which was also presented in this chapter. Furthermore, the differences in the dashboard implementation concerning the respective software stack could be shown. The stacks could be compared based on previously selected criteria in a utility analysis. The criteria were weighed against each other using a survey. The utility analysis clearly shows the superiority of stack 2 over the previous software. Finally, in this chapter, it was possible to design a sensor technology training within the framework of the LEAD Factory.



Figure 49. Finished setup in the LEAD-Factory

Conclusion

Finally, the activities and results of this work were used in Chapter 5 to answer the research questions defined at the beginning. Besides, the limitations of the two used systems were explained as well as the cooperation with the developers during the private beta phase.

6.2 Outlook

The implementation of the current status is a private beta version, i.e., there will still be many changes that have to be implemented in the context of the use of the factorycube, but also generate new possibilities. In the future, there are several possible further developments with the help of the factorycube.

With the new software stack, Industrial Analytics offers a new data model, the united manufacturing hub (UMH) MQTT data model. With this data model, ia: provides the user many basic data analytics functions. This applies to contextualized data, production data, and recommendations for action. To use these functions, the necessary data must be transferred to the ia: server via node-red. The UMH data model prescribes the data format and the necessary topic. Algorithms then process this data on the ia: server and finished evaluations such as the calculation of an OEE can be displayed in Grafana. At the time of the private beta, a large part of this data model was in progress and could be used in the future to expand the function of the factorycube in the LEAD Factory. Due to the open-source technology, it is also expected that additional functions will be included as yet unknown. In another use of the factorycube, you could also create templates yourself, which are made available to the open-source community, e.g., via GitHub. This would allow others to use them and participate in the development. Furthermore, it would also be possible for other learning factories to integrate a factorycube and its functions into their learning environments with little expenditure of time.

Another possibility of the factorycube in the vicinity of the learning factory is developing new use cases. Due to the extended functionality of data processing with the help of node-red, projects that were previously not possible can now be implemented. The integration of real-time inventory monitoring should be mentioned here. Currently, the goods or stocks of the LEAD-Factory are not monitored or recorded. By installing suitable sensors such as position sensors, light barriers, inductive and capacitive sensors, or weight sensors, the inventory can be tracked in real-time both in the central supermarket of the learning factory and at the workstations. This data could then be visualized and transferred to a possible ERP or automatic material ordering system.

Conclusion

With the answering of the research questions, implementing a digitized time measurement presented in this work is completed. Basically, this work and the knowledge gained with it result in new possibilities for improvements and future projects. A final conclusion can almost never be achieved with this. In conclusion, it can be said that the potential of using the factorycube in the environment of a learning factory is far from being exhausted and that this will not be the case in the future due to the constant further development of the product. The integration of the factorycube always just represents a snapshot of the current possibilities.

Bibliography

- [1] E. Koch, Globalisierung: Wirtschaft und Politik, München: Springer Fachmedien Wiesbaden 2014, 2017.
- [2] G. Ritzer and P. Dean, Globalization : The Essentials, Oxford: John Wiley & Sons, Incorporated, 2019.
- [3] H. Biedermann, S. Vorbach and W. Posch, Transformationen. Neue Wege zu industrieller Nachhaltigkeit, Augsburg, München: Rainer Hampp Verlag, 2017.
- [4] E. Abele and G. Reinhart, Zukunft der Produktion : Herausforderungen, Forschungsfelder, Chancen, München: Hanser, 2011.
- [5] C. Schlick, R. Bruder and H. Luczak, “Arbeitswirtschaft,” in *Arbeitswissenschaft*, Aachen, Darmstadt, Springer-Verlag GmbH Deutschland, 2017, pp. 551 - 601.
- [6] C. L. Niemeyer, I. Gehrke, K. Müller, D. Küsters and T. Gries, “Getting Small Medium Enterprises started on Industry 4.0 using,” RWTH Aachen University, Institut für Textiltechnik (ITA); McKinsey & Co.; ITA Academy GmbH, Aachen, 2020.
- [7] A. Meier and K. Michael, SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management, Fribourg, Rotkreuz: Springer Fachmedien Wiesbaden GmbH, 2019.
- [8] M. Hull, “Institut für Innovation und Industrie Management,” TU Graz, [Online]. Available:
<https://www.tugraz.at/institute/iim/forschung/forschungsprojekte/voladigital/>. [Accessed 26. 11. 2020].
- [9] IIM Institute, “IIM Institute,” [Online]. Available:
<https://www.tugraz.at/institute/iim/infrastruktur/lead-factory/>. [Accessed 07. 02. 2021].
- [10] E. Auberger, H. Karre and C. Ramsauer, “Introduction of a new product in an operating assembly process at Graz University of Technology's LEAD Factory,” Graz University of Technology, Institute of Innovation and Industrial Management, Graz, 2019.
- [11] A. Kumar, Personal, Academic and Career Development in Higher Education – SOA Ring to Success, London, New York: Taylor & Francis Group, 2007.
- [12] L. Kare, “A Rebuttal of NTL Institute's learning pyramid,” Lillehammer University College, Lillehammer, 2012.
- [13] K. Letrud and S. Hernes, “The diffusion of the learning pyramid myths in academia: an exploratory study,” in *Journal of Curriculum Studies*, Lillehammer, Routledge, 2015, pp. 291-302.

Bibliography

- [14] B. Ott, Grundlagen des beruflichen Lernens und Lehrens, Berlin: Cornelsen Verlag, 2017.
- [15] M. Steffen, S. Frye and J. Deuse, “Vielfalt Lernfabrik, Morphologie zu Betreibern, Zielgruppen und Ausstattungen von Lernfabriken im Industrial Engineering,” Technische Universität Dortmund, Dortmund, 2017.
- [16] T. Barton, C. Müller and C. Seel, “Wildauer Maschinen Werke – Digitale Lernfabrik für interdisziplinäre Lehre und Forschung,” in *Hochschulen in Zeiten der Digitalisierung*, Worms, Landshut, Wildau, Springer Fachmedien Wiesbaden GmbH, 2019, pp. 66-67.
- [17] J. Enke, M. Tisch and J. Metternich, “A guide to develop competency-oriented Lean Learning Factories systematically,” Technische Universität Darmstadt, TUprints, Darmstadt, 2016.
- [18] M. Tisch, C. Hertle, J. Cachay, E. Abele, J. Metternich and R. Tenberg, “A systematic approach on developing action-oriented, competency-based Learning Factories,” Elsevier B.V., Darmstadt, 2013.
- [19] E. Abelea, J. Metternicha, M. Tischa, G. Chryssolourisb, W. Sihnc, H. ElMaraghyd, F. Ranze and V. Hummele, “Learning Factories for research, education, and training,” in *5th Conference on Learning Factories 2015*, Graz, 2015.
- [20] A. Bothof and E. A. Hartmann, Zukunft der Arbeit in Industrie 4.0, Berlin: Springer Berlin Heidelberg, 2014.
- [21] J. Lee, B. Bagheri and H.-A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” Elsevier Ltd., Cincinnati, 2014.
- [22] P.-B. Bök, A. Noack, M. Müller and D. Behnke, Computernetze und Internet of Things, Datteln, Stralsund, Ostrhauderfehn, Dortmund: Springer Fachmedien Wiesbaden, 2019.
- [23] M. Ramgir, Internet of Things, Uttar Pradesh: Pearson India Education Service, 2019.
- [24] P. Alpar, F. Bensberg, R. Alt and P. Weimann, Anwendungsorientierte Wirtschaftsinformatik, Berlin, Leipzig, Marburg, Osnabrück: Springer Fachmedien Wiesbaden, 2019.
- [25] T. Kaufmann and H.-G. Servatius, Das Internet der Dinge und Künstliche Intelligenz als Game Changer, Walldorf, Düsseldorf, Stuttgart: Springer Fachmedien Wiesbaden, 2019.
- [26] “Learniot,” [Online]. Available: <http://www.learniot.com/cisco-model>. [Accessed 11. 02. 2021].
- [27] I. Gartner, “Gartner,” Gartner, Inc, [Online]. Available: <https://www.gartner.com/en/information-technology/glossary/big-data>. [Accessed 13. 02. 2021].

Bibliography

- [28] A. Meier and K. Michael, SQL- & NoSQL-Datenbanken, Fribourg, Rotkreuz: Springer-Verlag Berlin Heidelberg, 2016.
- [29] S. Zeisel, Big Data und Data Science in der strategischen Beschaffung, Mönchengladbach: Springer Fachmedien Wiesbaden GmbH, 2020.
- [30] C. Roberts, “Radio frequency identification (RFID),” *computers & security* 25, pp. 18 - 26, 2006.
- [31] R. Weinstein, “RFID: A Technical Overview and Its Application to the Enterprise,” *IT Pro*, pp. 27 - 33, May 2005.
- [32] C. Tribowski and G. Tamm, RFID, Berlin: Springer-Verlag Berlin Heidelberg, 2010.
- [33] Oracle, “Oracle Database,” Oracle Inc., [Online]. Available: <https://www.oracle.com/de/database/what-is-database/>. [Accessed 13. 02. 2021].
- [34] F. Herrmann, Datenorganisation und Datenbanken, Stuttgart: Springer Fachmedien Wiesbaden GmbH, 2018.
- [35] A. Meier, Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL & Co., Fribourg: Springer Fachmedien Wiesbaden GmbH, 2018.
- [36] P. Bühler, P. Schlaich and D. Sinner, Datenmanagement, Heidelberg: Springer-Verlag GmbH Deutschland, 2019.
- [37] “Digital Guide IONOS,” 1&1 IONOS SE, [Online]. Available: <https://www.ionos.at/digitalguide/server/knowhow/edge-computing-erklaerung-und-definition/>. [Accessed 04. 03. 2021].
- [38] J. Han, J. Du and E. Haihong, “Survey on NoSQL Database,” Beijing University of Posts and Telecommunications, Beijing, 2011.
- [39] “DNSstuff,” [Online]. Available: <https://www.dnsstuff.com/de/nosql-datenbankvergleich>. [Accessed 08. 03. 2021].
- [40] T. Dunning and E. Friedman, Time Series Databases: New Ways to Store and Access Data, Sebastopol, USA: O'Reilly Media, Inc., 2014.
- [41] “influxdata,” influxdata Inc., [Online]. Available: <https://www.influxdata.com/time-series-database/>. [Accessed 15. 02. 2021].
- [42] K. Kohler, “Datenarten und Datenverfügbarkeit,” Universität Ulm, Ulm, 2007.
- [43] D. Petrik, M. Mormul and P. Reimann, “Anforderungen für Zeitreihendatenbanken in der industriellen Edge,” in *HMD*, Stuttgart, Springer Fachmedien Wiesbaden GmbH, 2019, p. 1282–1308.
- [44] A. Bader, Comparison of Time Series Databases, Stuttgart: University of Stuttgart, 2015.
- [45] db-engines, “db-engines,” [Online]. Available: <https://db-engines.com/en/ranking/time+series+dbms>. [Accessed 15. 02. 2021].

Bibliography

- [46] K. Rudolph, A Comparison of NoSQL Time Series Databases, Berlin, 2015.
- [47] “TimescaleDB Overview,” Timescale Inc., [Online]. Available: <https://docs.timescale.com/latest/introduction>. [Accessed 06. 03. 2021].
- [48] S. Shabana and A. Danish, “Advanced Databases: Time Series Database with InfluxDB,” Ecole polytechnique de Bruxelles - ULB, Bruxelles, 2019.
- [49] “influxdata,” InfluxData Inc., [Online]. Available: <https://www.influxdata.com/>. [Accessed 18. 02. 2021].
- [50] S. Yfantidou and S. N. Z. Naqvi, “Time Series Databases and InfluxDB,” Universite libre de Bruxelles, Bruxelles, 2017.
- [51] P. Grzesik and D. Mrozek, “Comparative Analysis of Time Series Databases in the Context of Edge Computing for Low Power Sensor Networks,” in *Computational Science – ICCS 2020*, Gliwice, Springer Nature Switzerland AG, 2020, p. 371–383.
- [52] Timescale Inc., “Benchmarking TimescaleDB vs. InfluxDB for Time-Series Data,” Timescale Inc., 2020.
- [53] E. Schicker, Datenbanken und SQL: Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL, Regensburg: Springer Fachmedien Wiesbaden GmbH, 2016.
- [54] L. Jacobs and S. Hensel-Börner, “Die Kraft effektiver Daten Visualisierung – CLEAR(I): Ein Leitfaden zur wirkungsvollen Dashboard-Gestaltung,” in *Data-driven Marketing*, Hamburg, Springer Fachmedien Wiesbaden GmbH, 2020, pp. 43 - 75.
- [55] T. Kahl and F. Zimmer, Interaktive Datenvisualisierung in Wissenschaft und Unternehmenspraxis, Kamp-Lintfort: Springer Fachmedien Wiesbaden GmbH, 2020.
- [56] “Grafana,” Grafana Labs, [Online]. Available: <https://grafana.com/>. [Accessed 19. 02. 2021].
- [57] “Techopedia,” Techopedia Inc., 15. 06. 2017. [Online]. Available: <https://www.techopedia.com/definition/26418/data-analytics>. [Accessed 21. 02. 2021].
- [58] T. Runkler, Data Analytics: Models and Algorithms for Intelligent Data Analysis, München: Springer Fachmedien Wiesbaden GmbH, 2020.
- [59] M. Burkschat, E. Cramer and U. Kamps, Beschreibende Statistik: Grundlegende Methoden der Datenanalyse, Magdeburg, Aachen: Springer-Verlag Berlin Heidelberg, 2012.
- [60] G. Bourier, Beschreibende Statistik: Praxisorientierte Einführung, Wiesbaden: Betriebswirtschaftlicher Verlag Dr. Th. Gabler, 2008.

Bibliography

- [61] J. Hertzberg, K. Lingemann and A. Nüchter, Mobile Roboter: Eine Einführung aus Sicht der Informatik, Osnabrück: Springer-Verlag Berlin Heidelberg, 2011.
- [62] W. Stier, Methoden der Zeitreihenanalyse, St. Gallen: Springer-Verlag Berlin Heidelberg, 2001.
- [63] F. Klinker, "Exponential moving average versus moving exponential average," in *Mathematik in Forschung und Anwendung*, Dortmund, Springer-Verlag, 2010, pp. 97 - 107.
- [64] Institut für Industriebetriebslehre und Industrielle Produktion, "REFA - Eine Zeitstudie," Universität Karlsruhe, Karlsruhe.
- [65] M. Röhrig, Methodische Arbeitsplatz- und Prozessanalyse in der Akkordarbeit, Landshut: Universitätsverlag Ilmenau, 2015.
- [66] A. Frühwald, Das REFA-Gedankengut, Karlsruhe: Springer Fachmedien Wiesbaden GmbH, 1955.
- [67] R. B. e.V., "REFA Bundesverband e.V. und REFA AG," [Online]. Available: <https://refa.de/refa/refa-group/refa-bundesverband-e-v-und-refa-ag>. [Accessed 17. 02. 2021].
- [68] Refa, "REFA-Zeitstudie," in *Industrial Engineering: Standardmethoden zur Produktivitätssteigerung und Prozessoptimierung*, Darmstadt, Hanser Fachbuchverlag, 2015, pp. 187 - 193.
- [69] G. Sandhaus, B. Berg and P. Knott, Hybride Softwareentwicklung, Düsseldorf: Springer-Verlag Berlin Heidelberg, 2014.
- [70] "Rechner Sensors," Rechner Industrie-Elektronik GmbH, [Online]. Available: <https://www.rechner-sensors.com/dokumentation/wissen/io-link-einfach-verstaendlich-erklaert>. [Accessed 22. 02. 2021].
- [71] "industrial-analytics," ia: industrial analytics GmbH, [Online]. Available: <https://www.industrial-analytics.net/de/>. [Accessed 24. 02. 2021].
- [72] S. Gotthardt, Simulation of an RFID-aided and Digitized Milk-run Logistics System within the Context of a Learning Factory, Graz: Graz University of Technology, 2018.
- [73] J. Kühnapfel, "Das Vorgehen bei der Nutzwertanalyse," in *Nutzwertanalysen in Marketing und Vertrieb*, Ludwigshafen am Rhein, Springer Fachmedien Wiesbaden GmbH, 2019, pp. 5-21.
- [74] "Node-RED," OpenJS Foundation, [Online]. Available: <https://nodered.org/>. [Accessed 15. 04. 2021].
- [75] S. I. Tay, L. T. Chuan, A. H. N. Aziati and A. N. Aizat, "An Overview of Industry 4.0: Definition, Components, and Government Initiatives," *Journal of Advanced Research in Dynamical and Control Systems*, pp. 1379-1387, 2018.

Bibliography

- [76] I. Xie and K. Matusiak, “Sciencedirect,” [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/metadata>. [Accessed 06. 03. 2021].
- [77] “Einführung in JSON,” [Online]. Available: <https://www.json.org/json-de.html>. [Accessed 21. 05. 2021].
- [78] “Digital Guide IONOS,” [Online]. Available: <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklaert/>. [Accessed 21. 05. 2021].

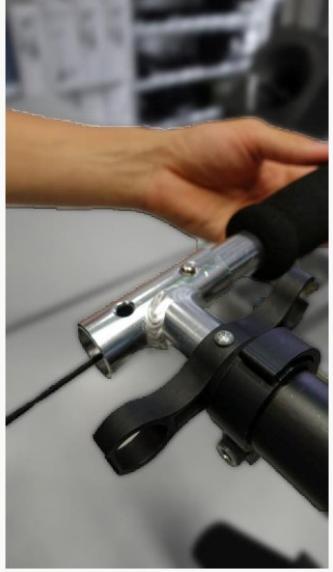
Appendix

Appendix A: Work Instruction

| | | |
|-----|--|---|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 1 out of 12 |
| | | IIM Bold Wheel Variant A : The Standard Version A 07/2019 F. Mast  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|---|--|
| 1 | <ul style="list-style-type: none"> - grab Item 5: Handlebar w/ Grip with right hand and Item 3: Handlebar Cap w/ Cord with left hand - with left hand drop the cord through the end of the handlebar while holding handlebar with right hand  |
| 2 | <ul style="list-style-type: none"> - push end of the cord into the handlebar with left hand - push the handlebar cap into end of handlebar with left hand while holding handlebar with right hand  |

| | | |
|---|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 2 out of 12 |
| IIM Bold Wheel Variant A : The Standard | | |
| Version A 07/2019 | F. Mast |  INNOVATION AND INDUSTRIAL MANAGEMENT |

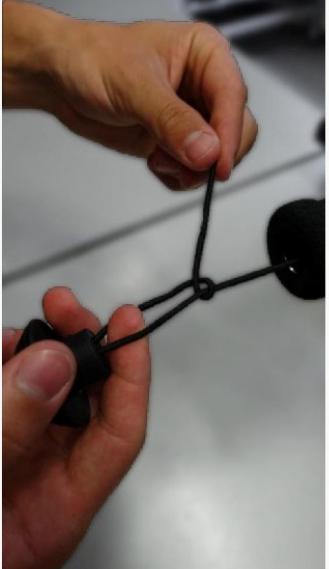
| # | Task |
|---|---|
| 3 | <ul style="list-style-type: none"> – with right hand, rotate steering tube so the holes in the T-Bar are facing up – pull the cord through the T-Bar with the right hand while holding handlebar with left hand  |
| 4 | <ul style="list-style-type: none"> – with left hand, press the button in and insert the handlebar into the T-Bar and align the button with the hole in the T-Bar – pull the cord tight with the right hand to prevent tangles  |

| | | |
|-----|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 3 out of 12 |
| | IIM Bold Wheel Variant A : The Standard |  INNOVATION AND INDUSTRIAL MANAGEMENT |

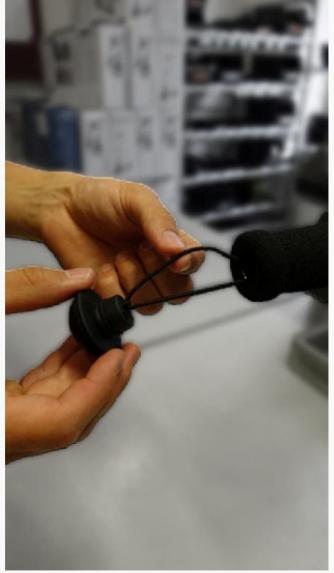
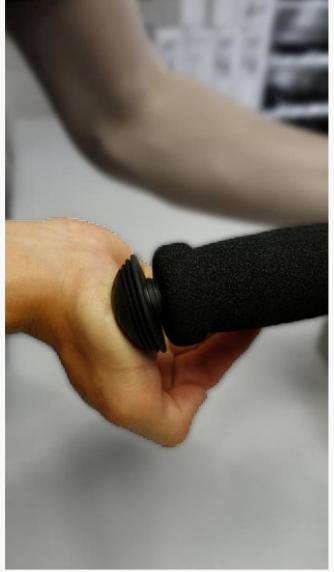
| # | Task |
|---|--|
| 5 | <ul style="list-style-type: none"> – grab Item 5: Handlebar w/ Grip with right hand – guide the cord through the handlebar with the left hand – with right hand, press the button in and insert the handlebar into the T-Bar and align the button with the hole in the T-Bar |
| 6 | <ul style="list-style-type: none"> – grab Item 2: Handlebar Cap with right hand – with left hand, guide the cord through the end of the handlebar cap while holding the handlebar cap with right hand |



| | | |
|-----|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 4 out of 12 |
| | IIM Bold Wheel Variant A : The Standard |  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|---|---|
| 7 | <ul style="list-style-type: none"> – with left hand, guide the cord back through the handlebar cap while holding the handlebar cap with right hand  |
| 8 | <ul style="list-style-type: none"> – with left hand, stretch the cord and hold the handlebar cap 100mm from the end of the handlebar – with right hand, tie a knot in the cord  |

| | | |
|-----|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 5 out of 12 |
| | IIM Bold Wheel Variant A : The Standard |  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|----|---|
| 9 | <ul style="list-style-type: none"> – with right hand, push the end of the cord inside the handlebar, while holding the handlebar cap with left hand  |
| 10 | <ul style="list-style-type: none"> – with right hand, push the handlebar cap into the end of the handlebar while holding the handlebar with left hand  |

| | | |
|-----|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 6 out of 12 |
| | IIM Bold Wheel Variant A : The Standard |  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|----|--|
| 11 | <ul style="list-style-type: none"> – with right hand, remove the handlebar and place in the holder clip on the T-Bar while holding it with left hand – do this for both handlebars |
| 12 | <ul style="list-style-type: none"> – using both hands, reposition the entire assembly onto its side so the holes in the frame are facing up |



Step: 7 out of 12

WP4

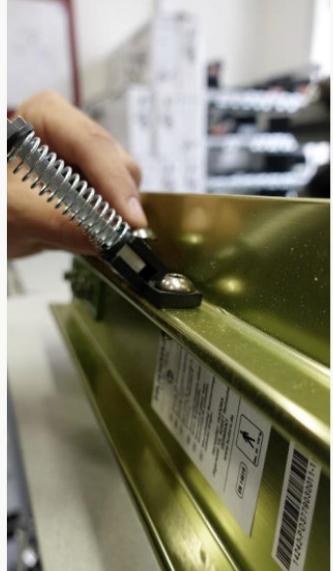
Work Item:
**Rear Wheel, Kickstand
& Assembly**

IIM Bold Wheel Variant A : The Standard

Version A
07/2019

F. Mast

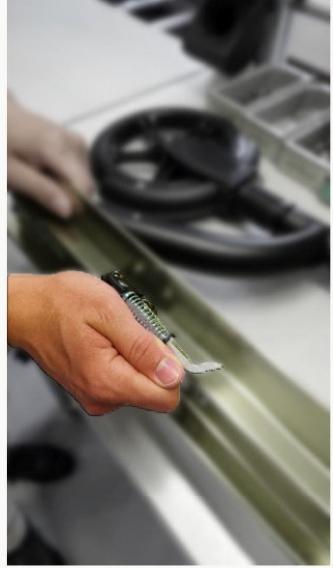
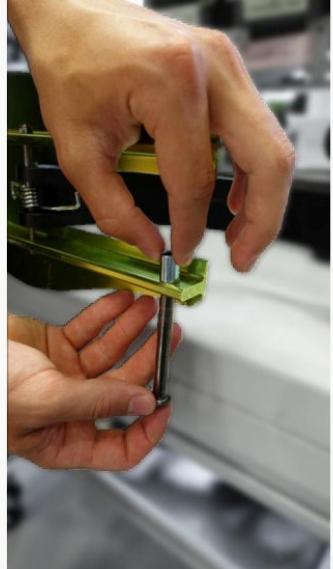


| # | Task |
|----|--|
| 13 | <ul style="list-style-type: none"> – grab Item 1: Kickstand with right hand and place over holes in frame  |
| 14 | <ul style="list-style-type: none"> – grab Item A: M6x18 Capscrew with left hand and place into hole of kickstand while holding the <i>kickstand</i> (it) with right hand – repeat for second hole – release kickstand from right hand  |

| | | |
|-----|---|---|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 8 out of 12 |
| | IIM Bold Wheel Variant A : The Standard Version A 07/2019 | F. Mast  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|----|---|
| 15 | <ul style="list-style-type: none"> – grab Item B: M6 Lock Nut with right hand and hand tighten onto capscrew while holding capscrew head with left hand – repeat for second screw  |
| 16 | <ul style="list-style-type: none"> – grab wrench with left hand and place onto nut and hold – grab drill with right hand and insert into head of capscrew – put wrench and drill into storage position  <p style="text-align: center;">⚠ Drill must be in the #1 position</p> |

| | | |
|-----|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 9 out of 12 |
| | IIM Bold Wheel Variant A : The Standard | Version A 07/2019 F. Mast  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|----|--|
| 17 | <ul style="list-style-type: none"> – with right hand, put kickstand into the closed position  |
| 18 | <ul style="list-style-type: none"> – grab <u>Item E: Ø6x70 Shoulder Bolt</u> with right hand – grab <u>Item D: Ø8x15 Spacer</u> with left hand – insert shoulder bolt into hole from bottom side with right hand – slide spacer onto bolt with left hand – with right hand, slide bolt so it is flush with the end of the spacer  |

Step: 10 out of 12

WP4

Work Item:
Rear Wheel, Kickstand & Assembly

IIM Bold Wheel Variant A : The Standard

Version A
07/2019

F. Mast



| # | Task |
|----|--|
| 19 | <ul style="list-style-type: none"> – grab Item 4B Std Wheel (Black) – 180mm with left hand – align the wheel bearing with the end of the bolt <p>! Attention: Text on the wheel must face upwards</p> |
| 20 | <ul style="list-style-type: none"> – grab Item D: Ø8x15 Spacer and place over the hole of the wheel bushing while holding the shoulder bolt with the right hand – grab the punch with left hand and push through the top hole and the wheel bearing while holding shoulder bolt with right hand |



| | | |
|-----|---|---|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 11 out of 12 |
| | IIM Bold Wheel Variant A : The Standard Version A 07/2019 | F. Mast  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|----|---|
| 21 | <ul style="list-style-type: none"> – push the shoulder bolt upwards with the right one hand while aligning the spacers and wheel bearings using the punch in the other left hand  |
| 22 | <ul style="list-style-type: none"> – grab Item C: M6x12 Capscrew with left hand and insert into the end of the shoulder bolt and hand tighten while holding the head of the capscrew with right hand – release capscrew and grab drill with right hand – insert drill into head of capscrew and tighten capscrew until drill clicks while holding the head of the capscrew with left hand <p>Attention: drill must be in the #1 position</p> <ul style="list-style-type: none"> – place drill back into storage position   <p>Drill must be in the #1 position</p> |

| | | |
|-----|---|--|
| WP4 | Work Item: Rear Wheel, Kickstand & Assembly | Step: 12 out of 12 |
| | IIM Bold Wheel Variant A : The Standard |  INNOVATION AND INDUSTRIAL MANAGEMENT |

| # | Task |
|----|--|
| 23 | <ul style="list-style-type: none"> – with both hands, reposition scooter upright and with the front wheel to the left side <div style="text-align: center; margin-top: 20px;">  </div> |
| | <div style="display: flex; justify-content: space-between; width: 100%;"> Last update: 16.07.2019 by: Mast Cycle Time: 195 sec </div> |

Appendix

Appendix B: Software Stack 1 (Influxdata)

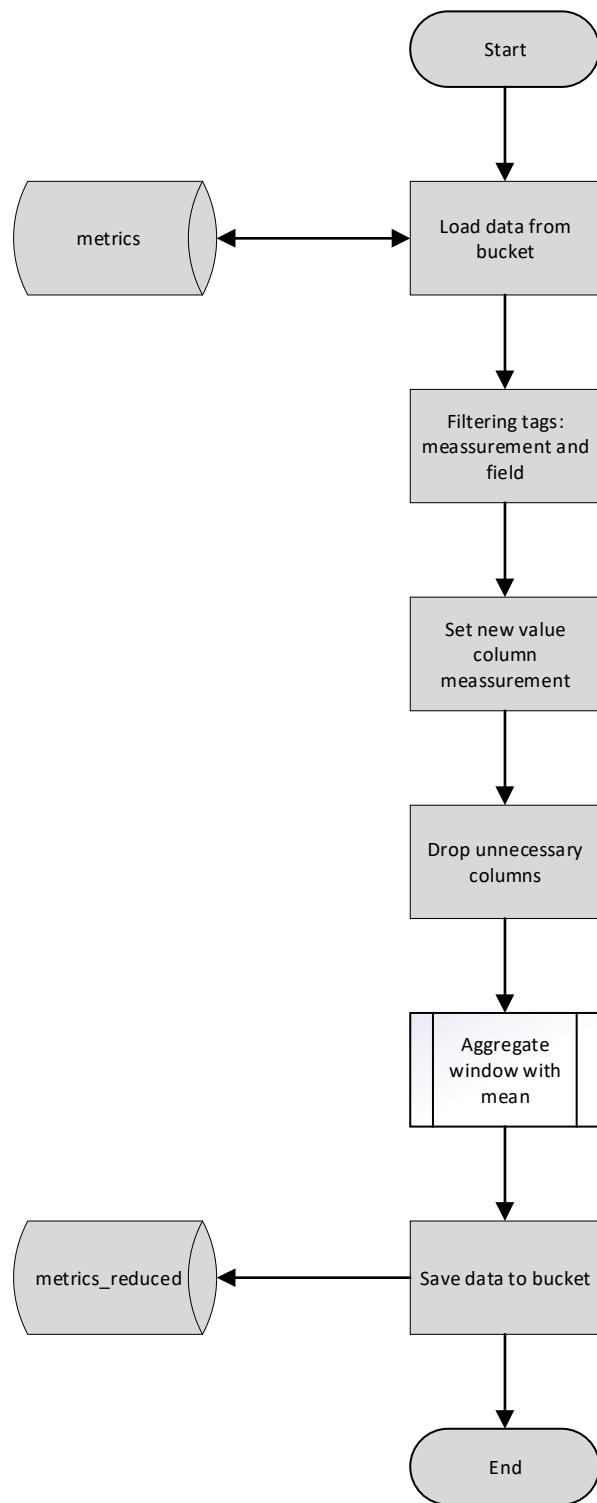


Figure 50. Task 1 downsampling step 1

Appendix

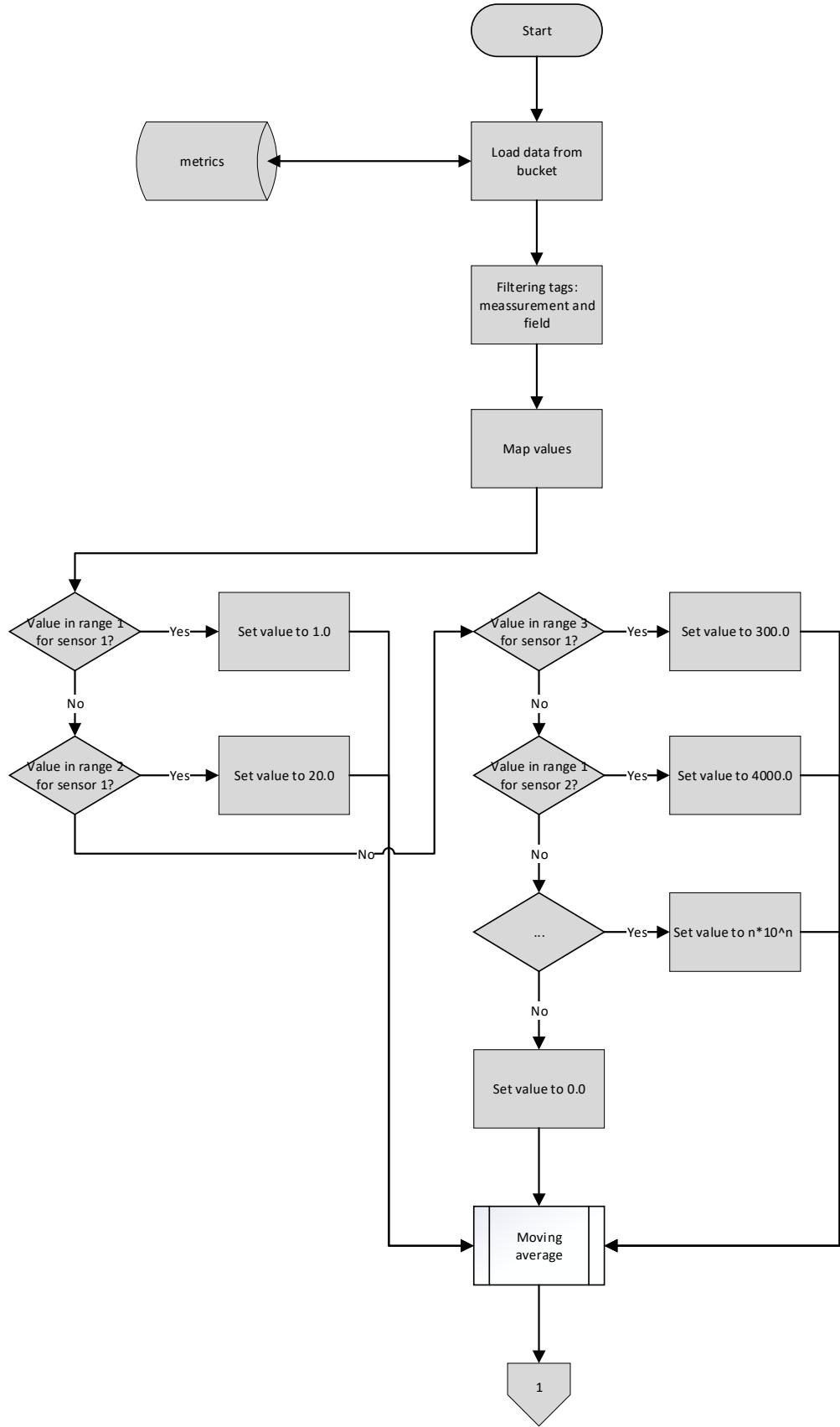


Figure 51. Task 2-1 processing position sensor

Appendix

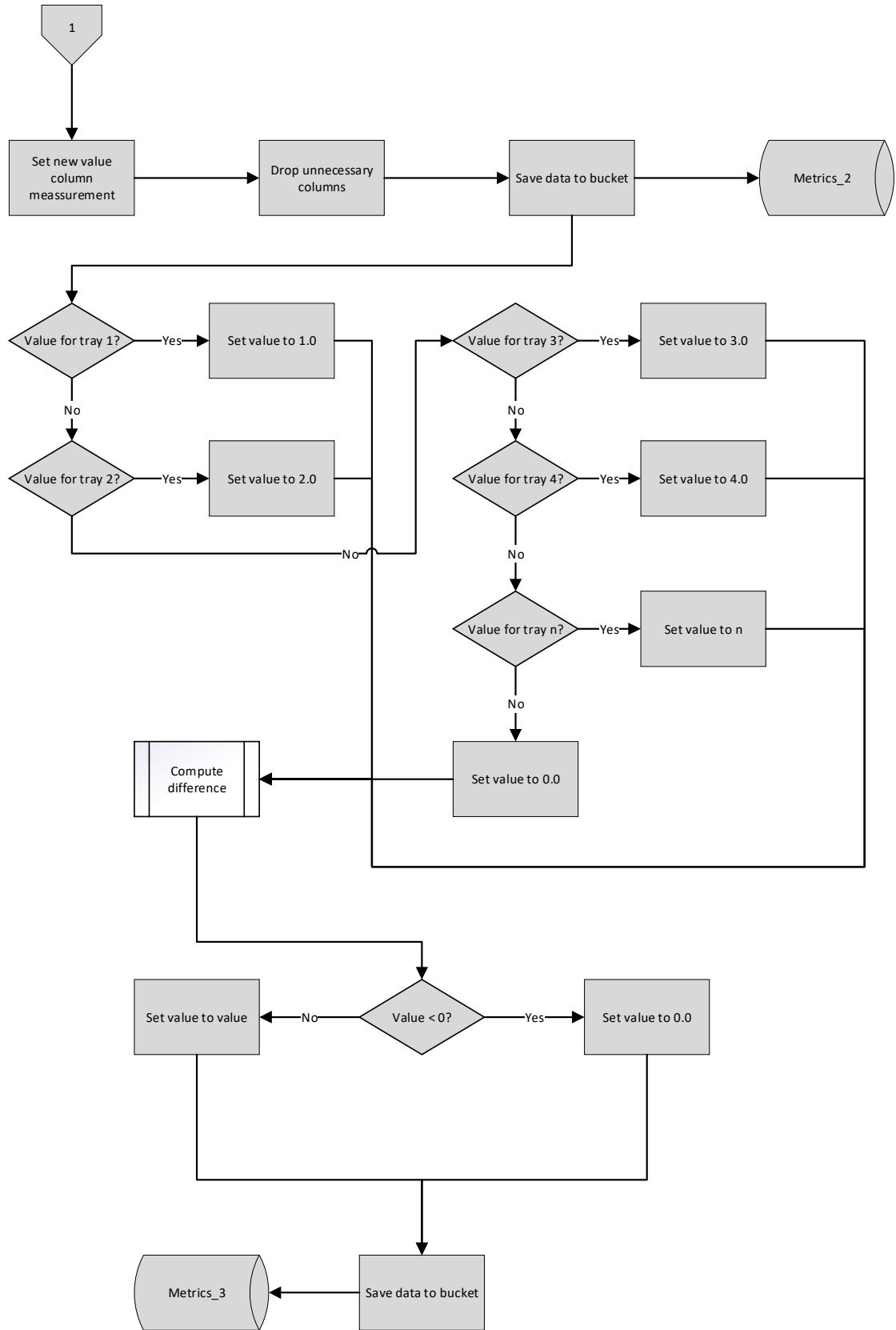


Figure 52. Task 2-2 processing position sensor

Appendix

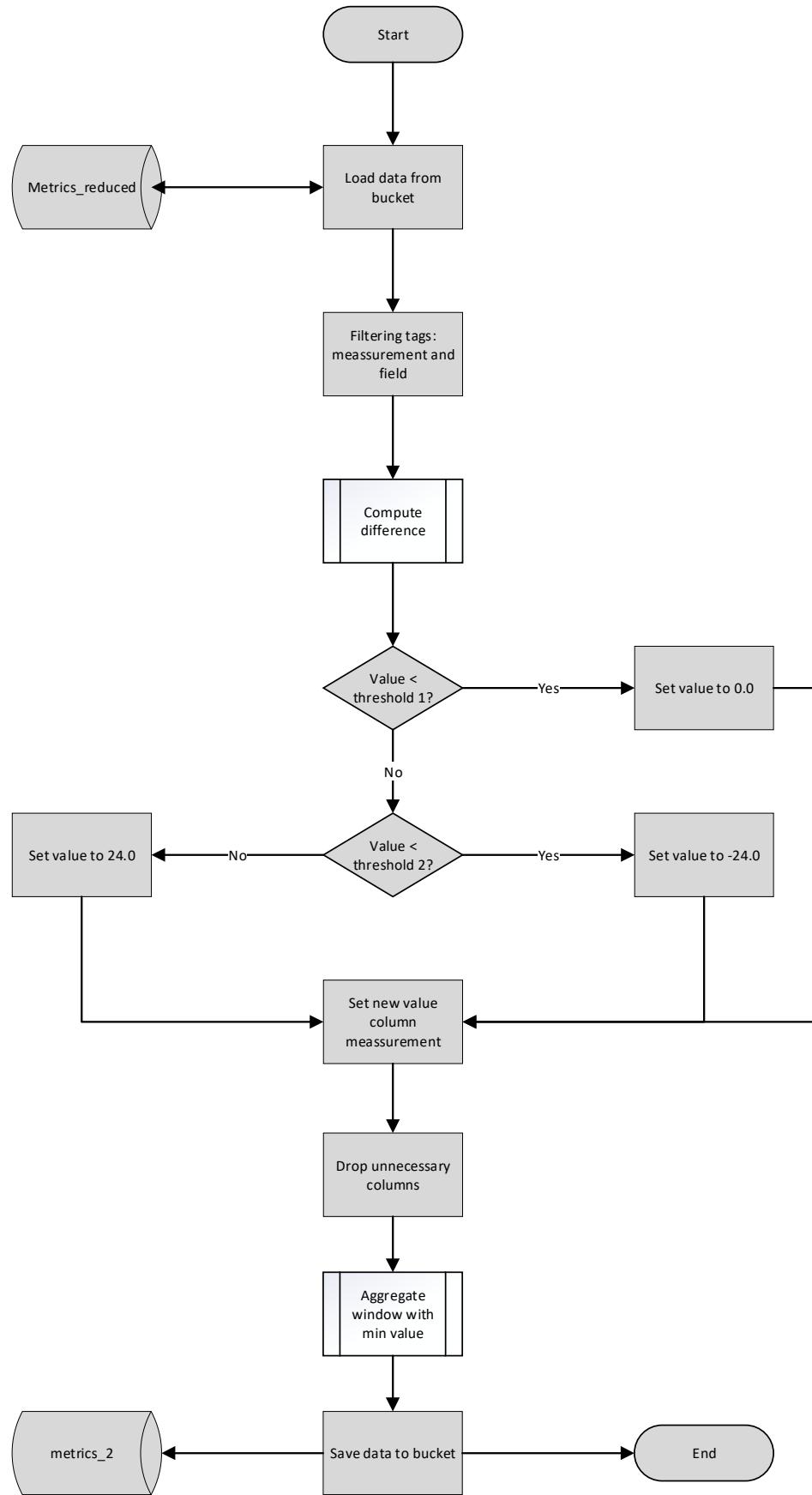


Figure 53. Task 3 processing vibration sensor

Appendix

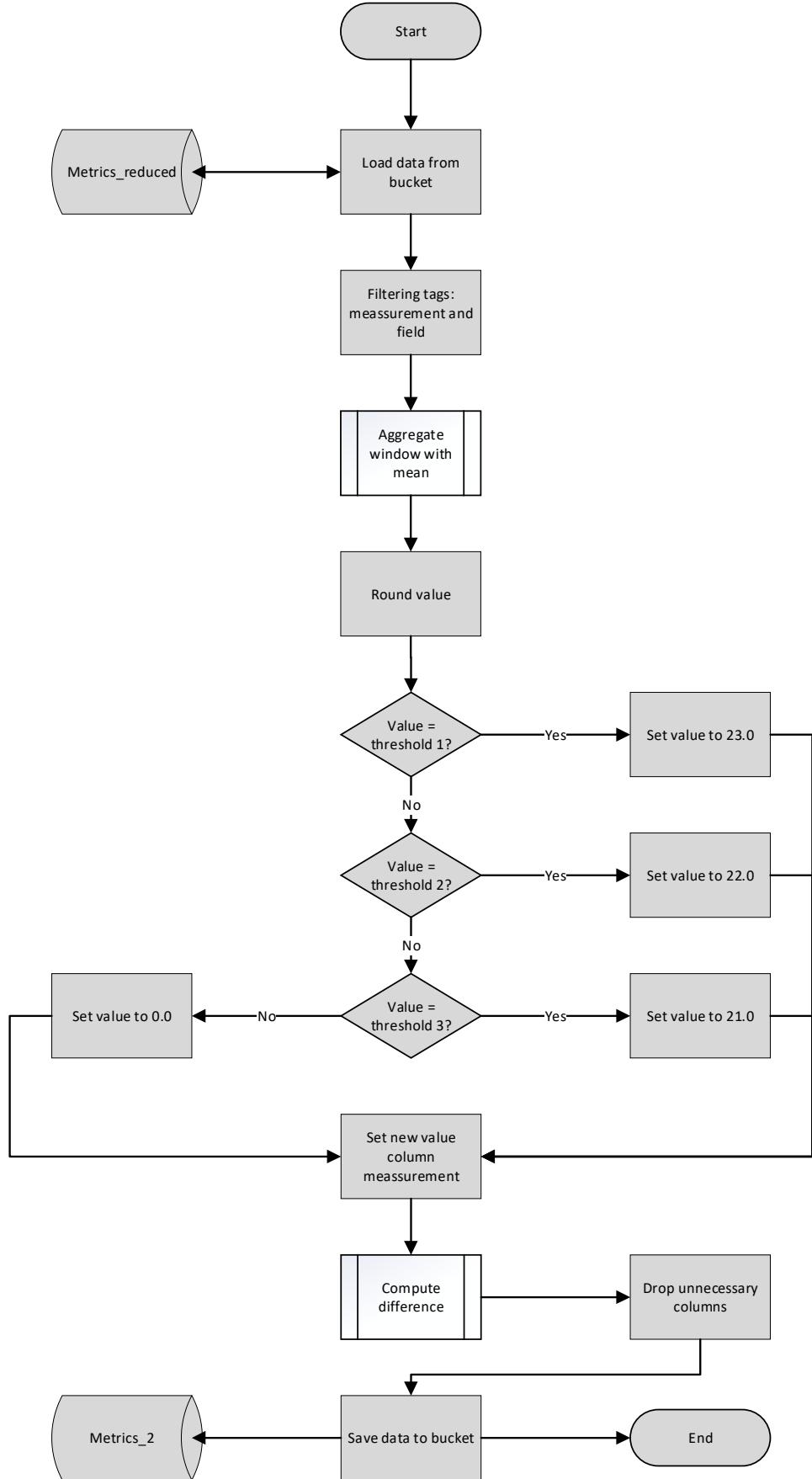


Figure 54. Task 4 processing induction sensor

Appendix

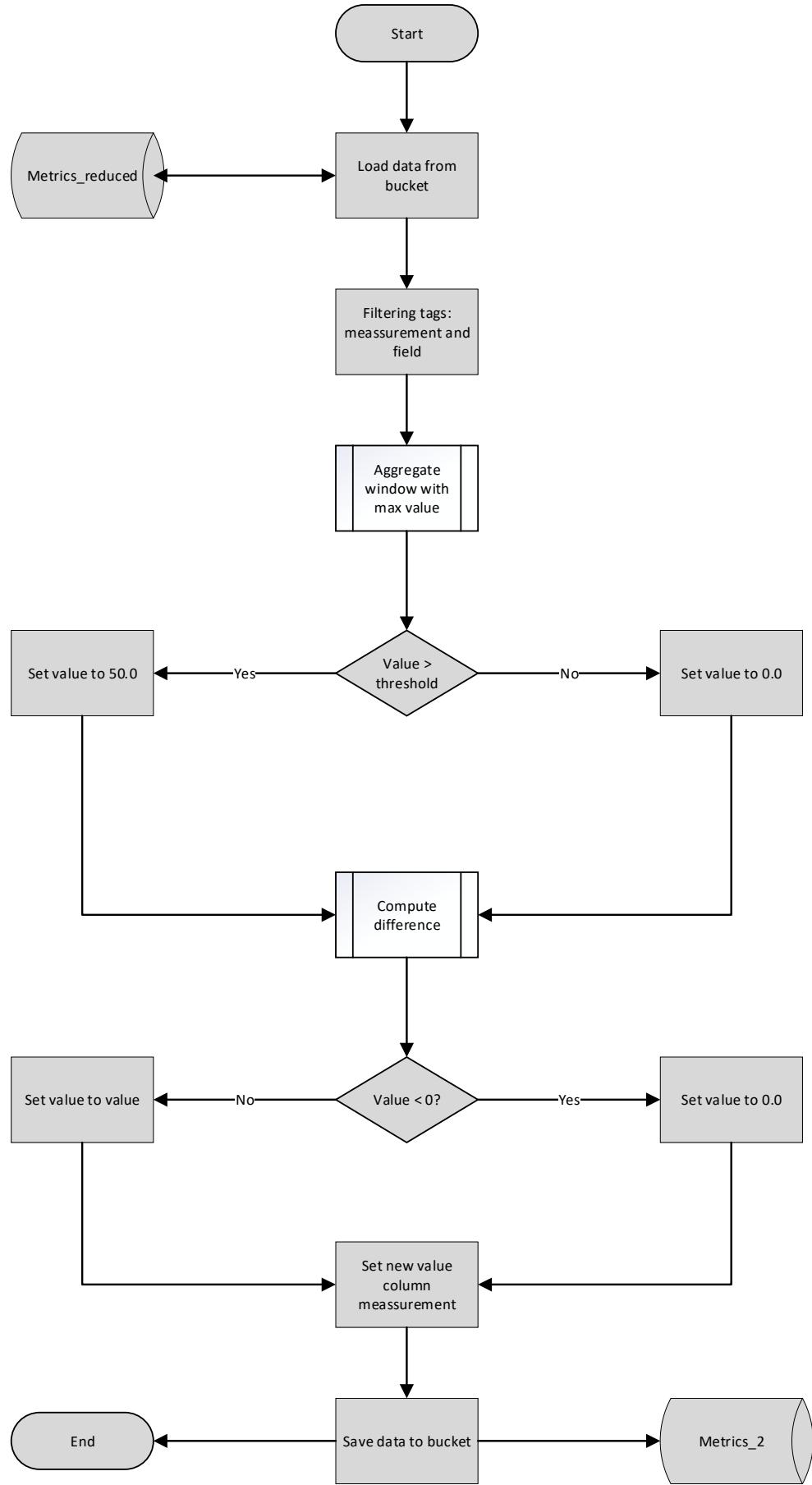


Figure 55. Task 5 processing PMD profiler

Appendix

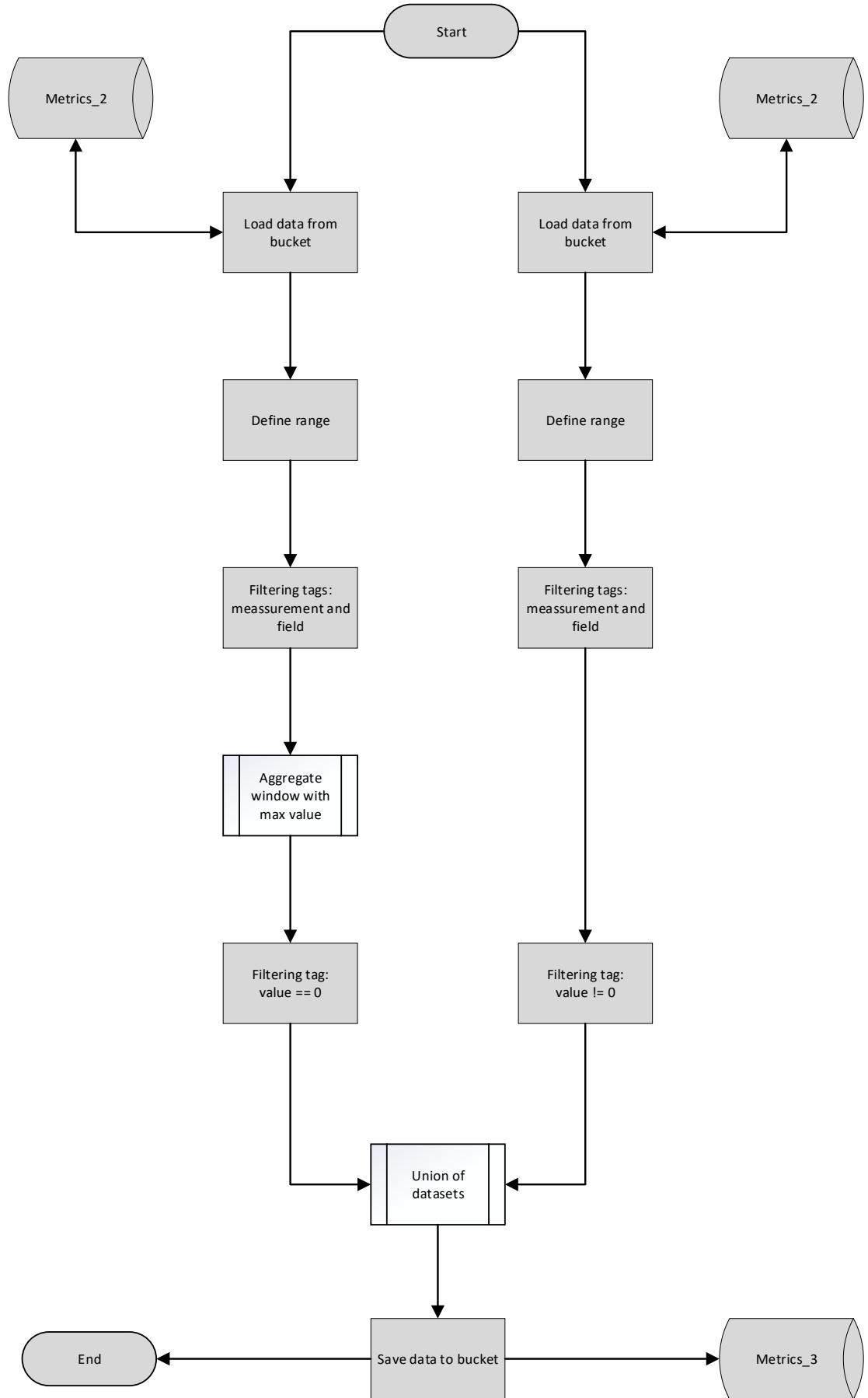


Figure 56. Task 2, 3, 4, 5 downsampling

Appendix

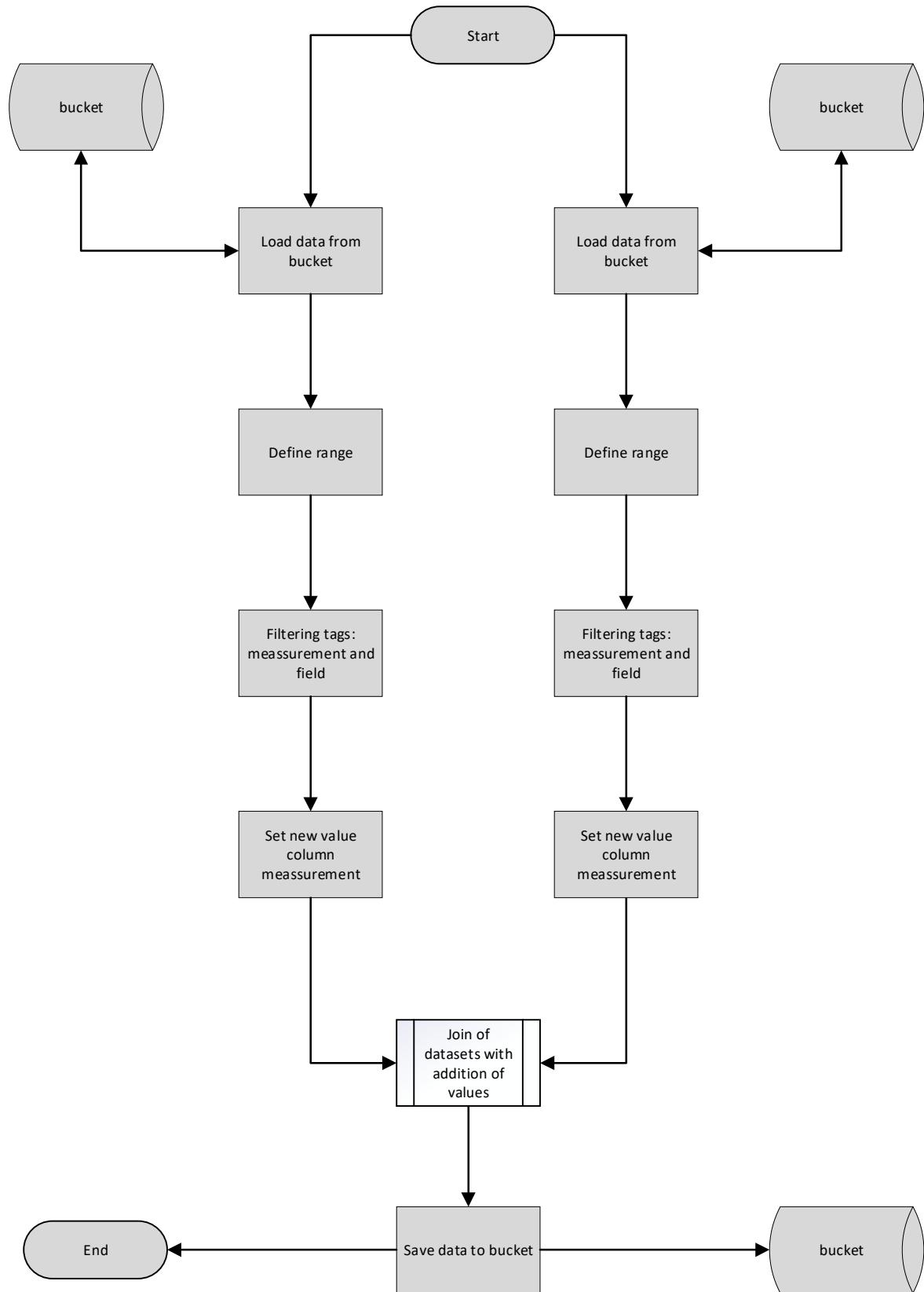


Figure 57. Task 6 to 13 join/union tables

Appendix

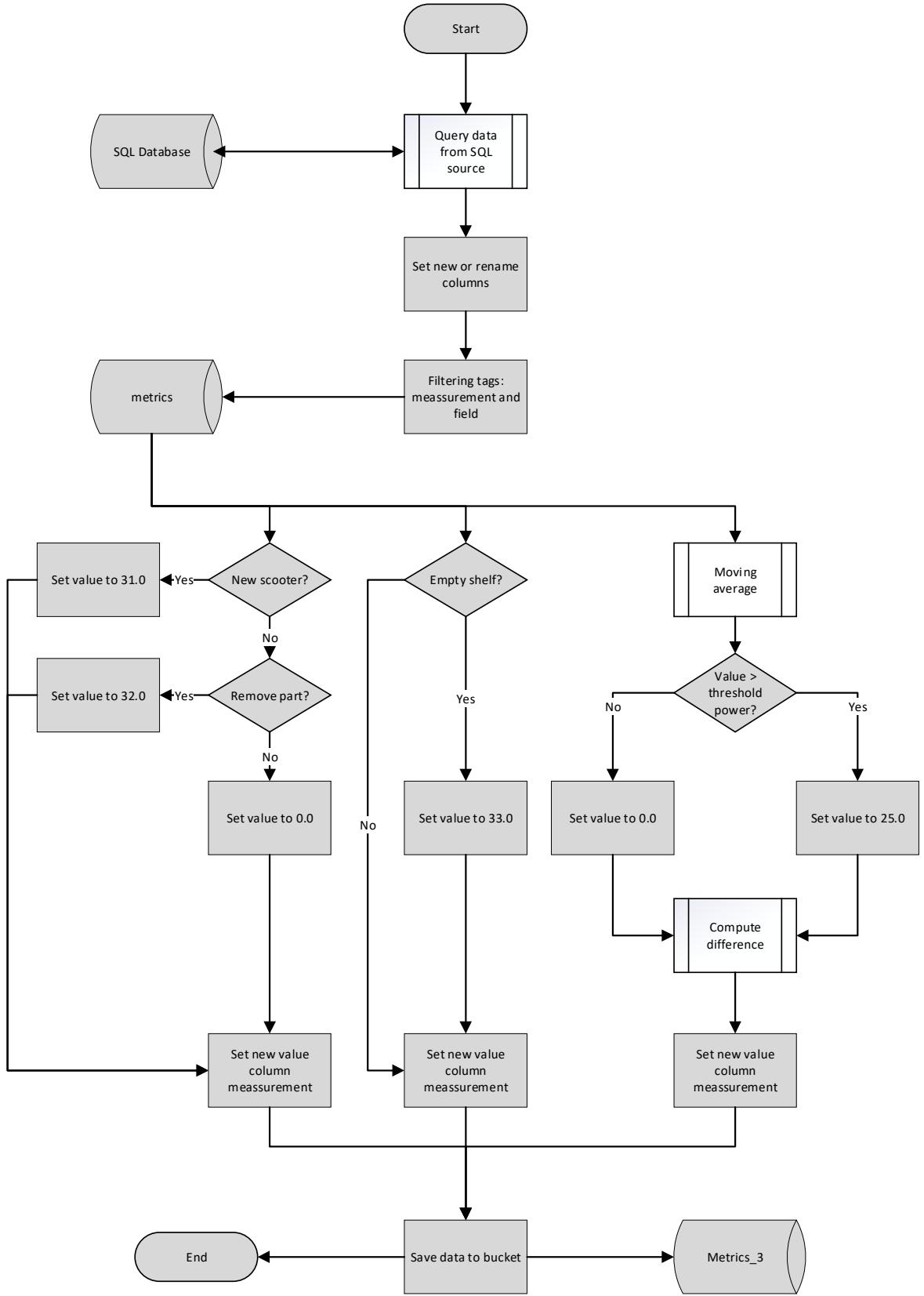


Figure 58. Task 14 reading and processing of the LEAD Factory data

Appendix

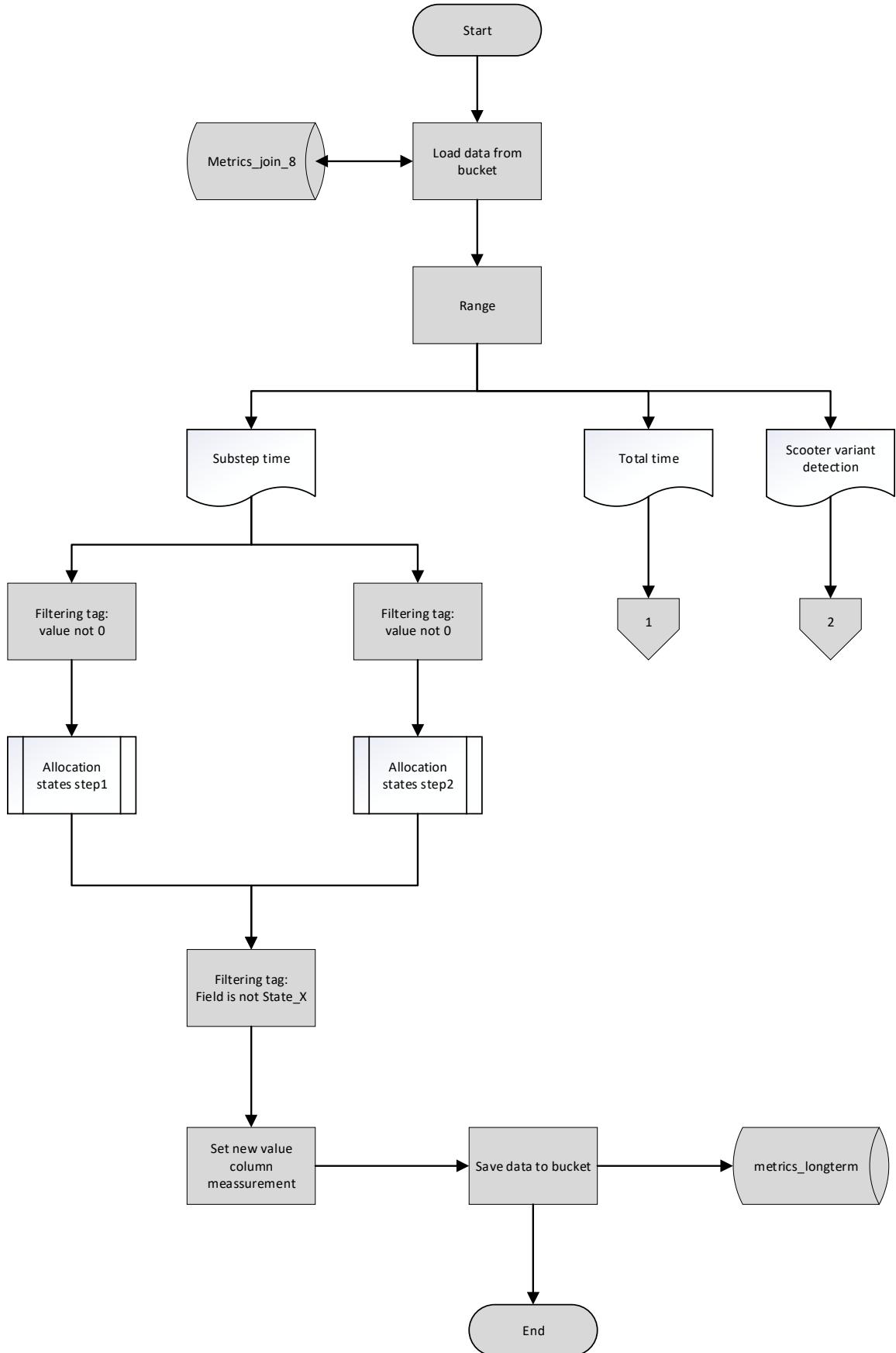


Figure 59. Task 15 processing longterm data 1

Appendix

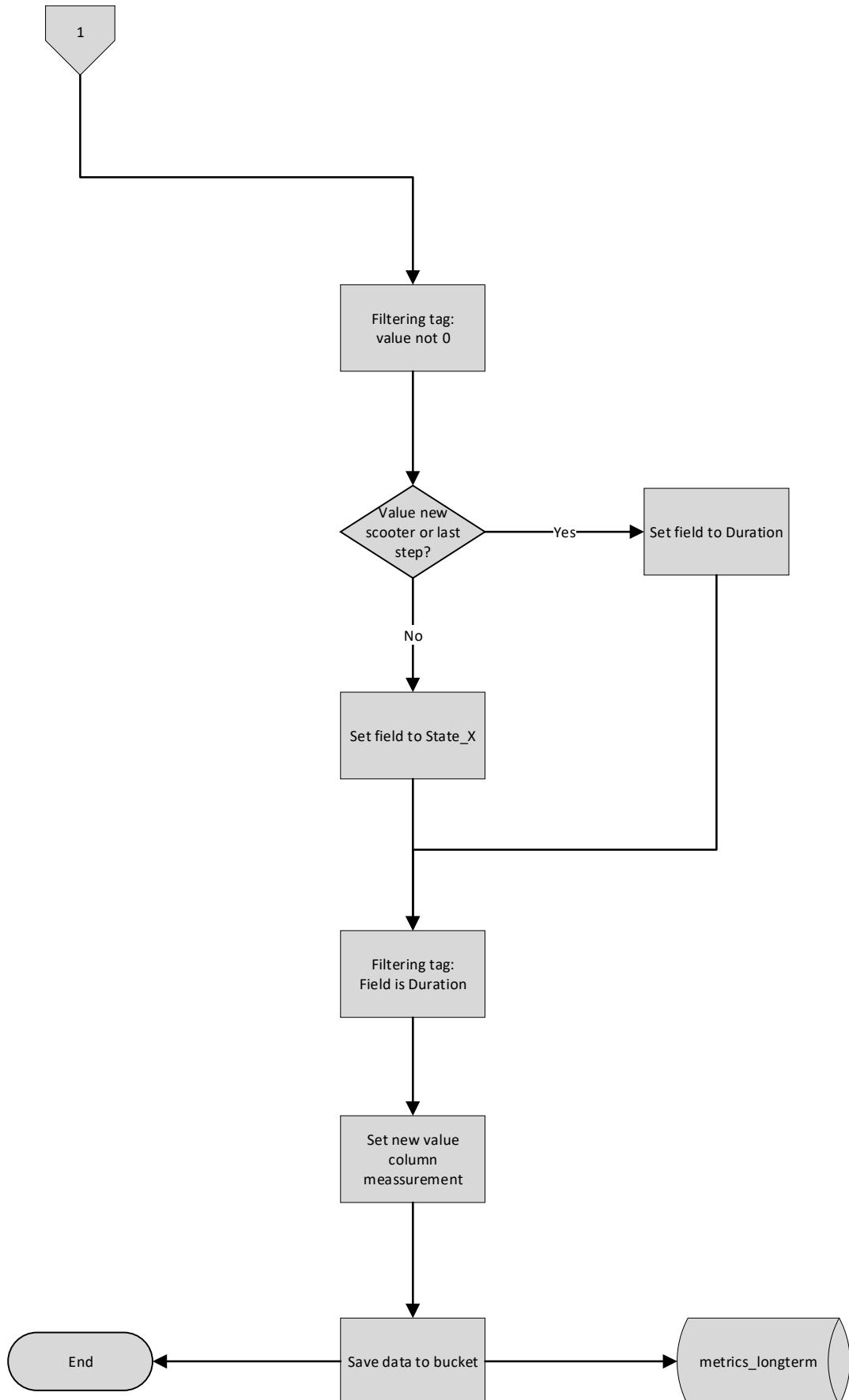


Figure 60. Task 15 processing longterm data 2

Appendix

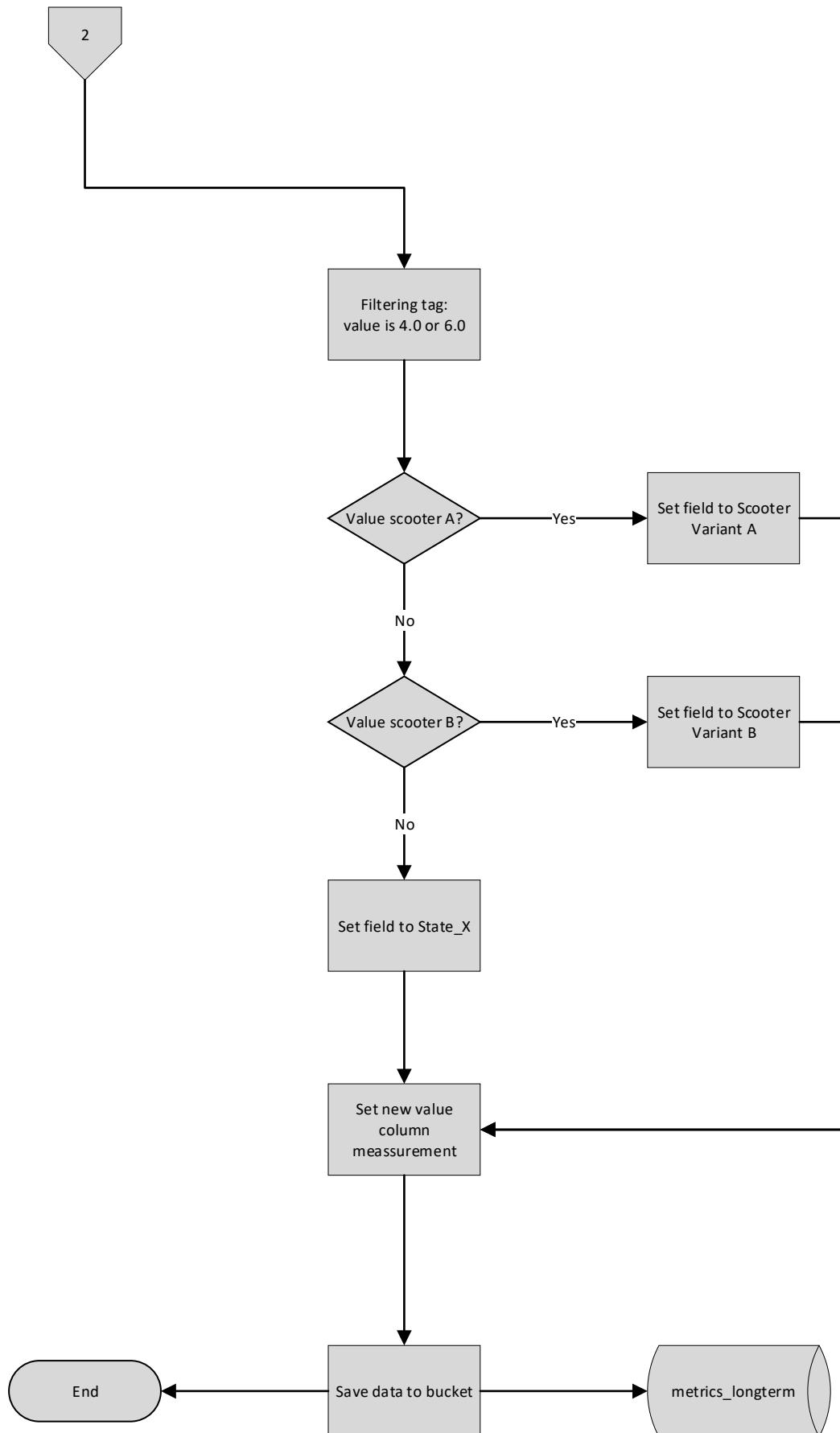


Figure 61. Task 15 processing longterm data 3

Appendix

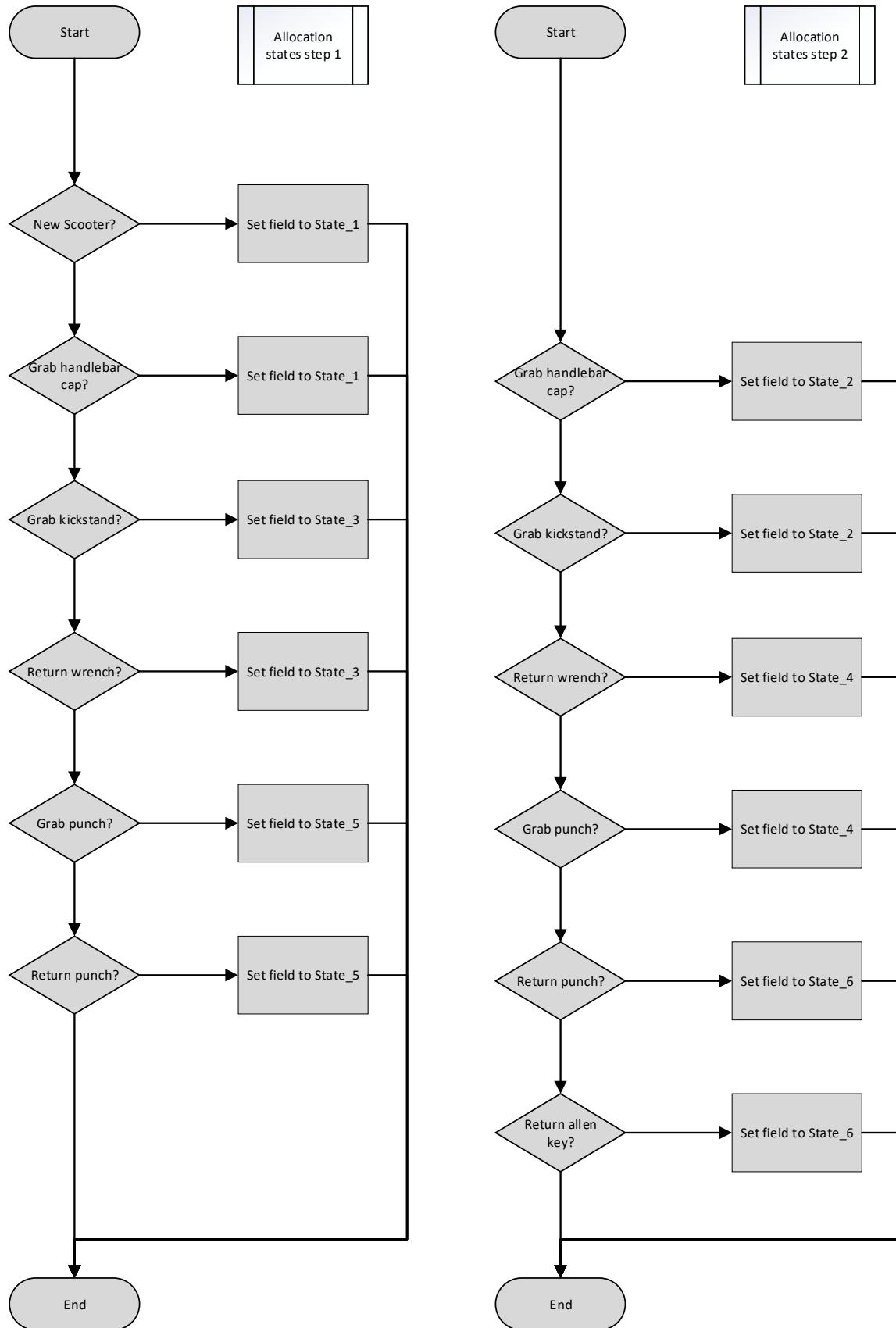


Figure 62. Task 15 substep processing longterm data 4

Appendix

Appendix C: Software Stack 2 (Node-red and TimescaleDB)

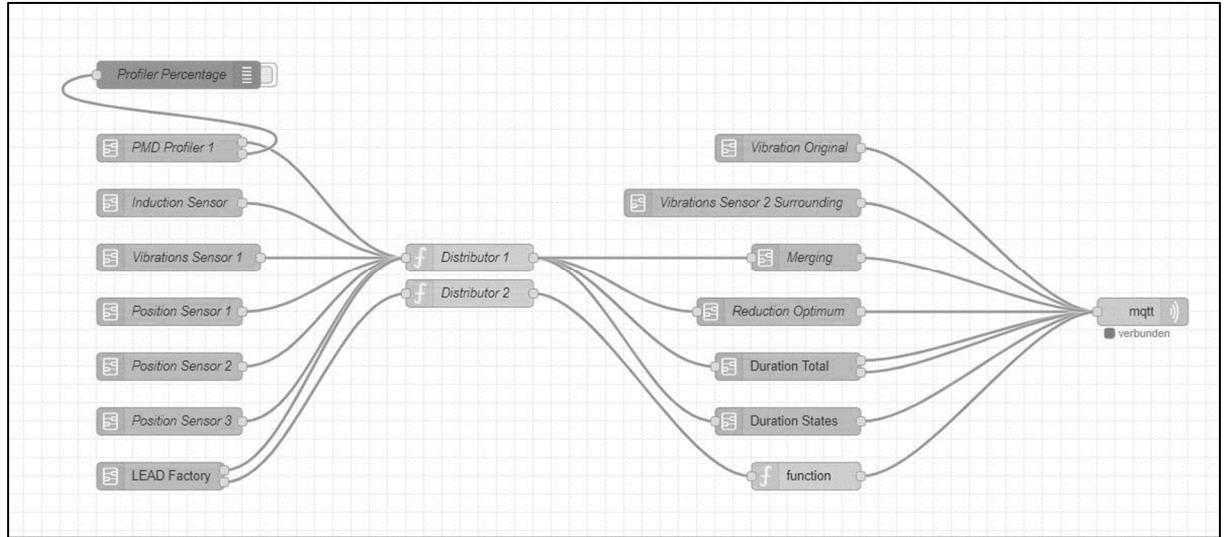


Figure 63. Main data processing implemented in node-red

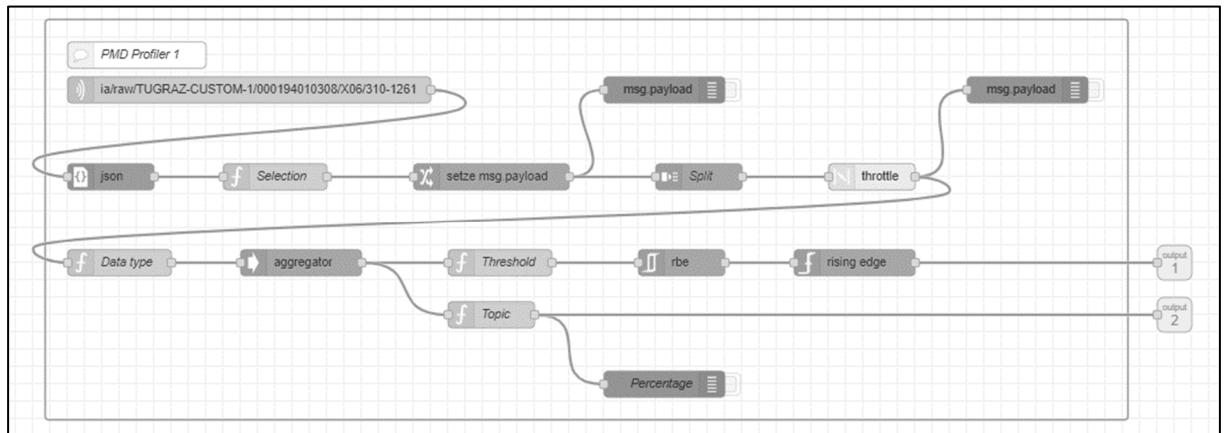


Figure 64. Data processing PMD profiler

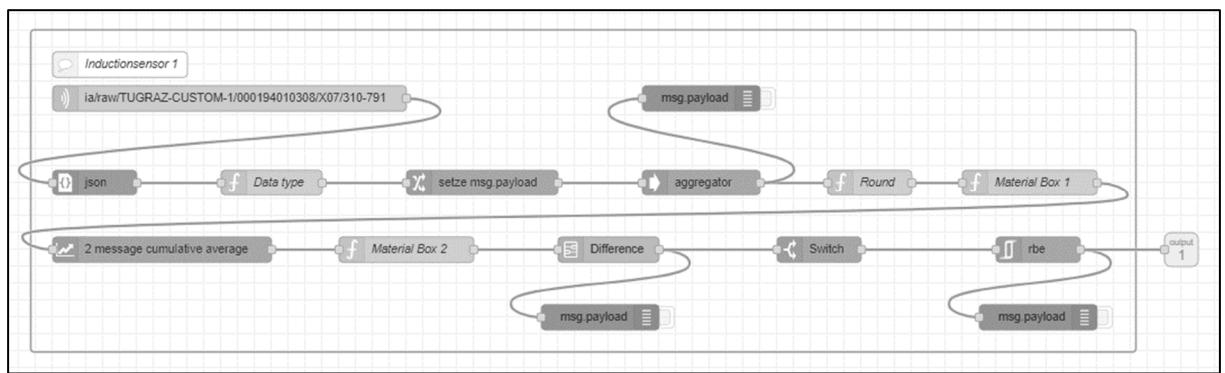


Figure 65. Data processing induction sensor

Appendix

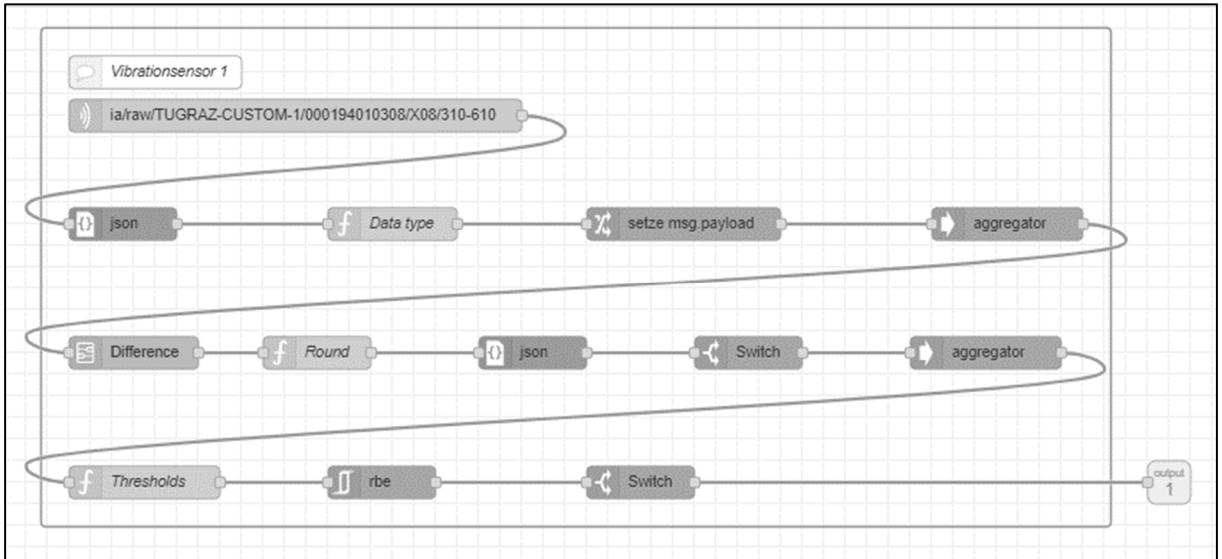


Figure 66. Data processing vibration sensor

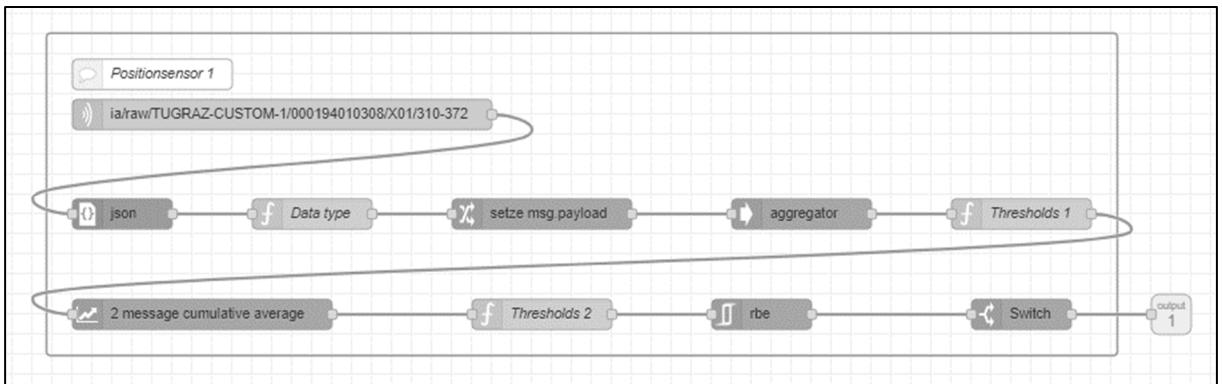


Figure 67. Data processing position sensor

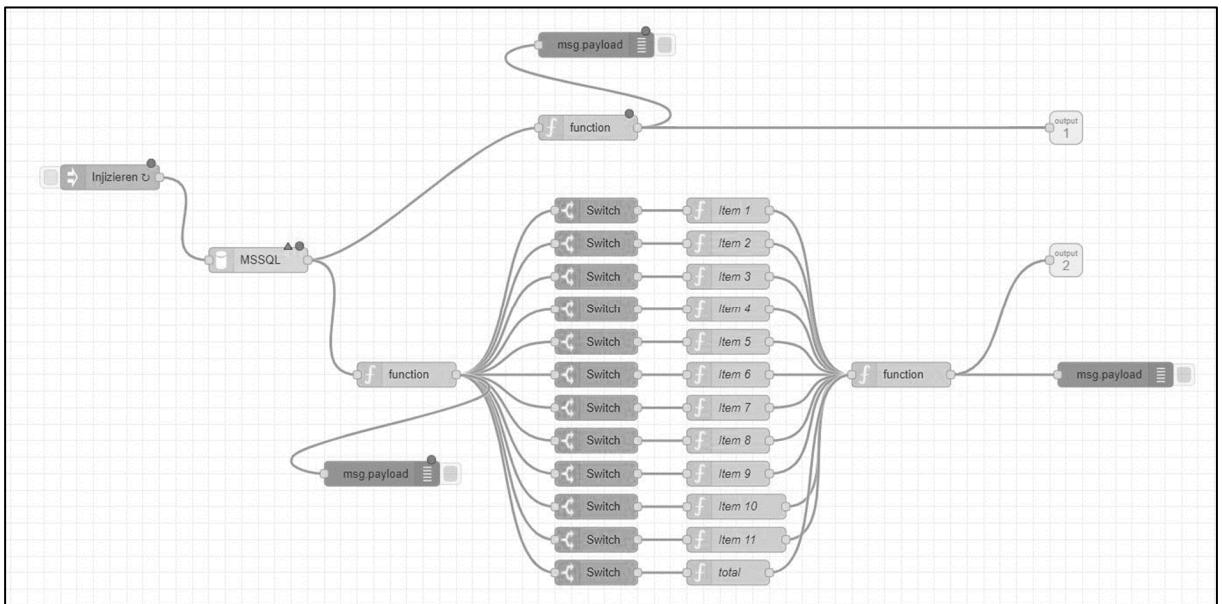


Figure 68. Reading and processing of the LEAD Factory data

Appendix

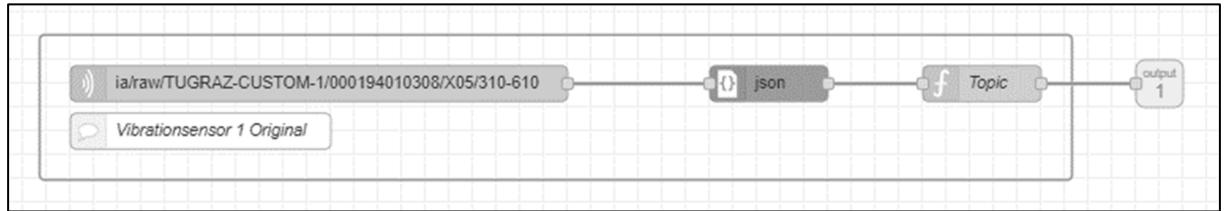


Figure 69. Data processing vibration sensor original

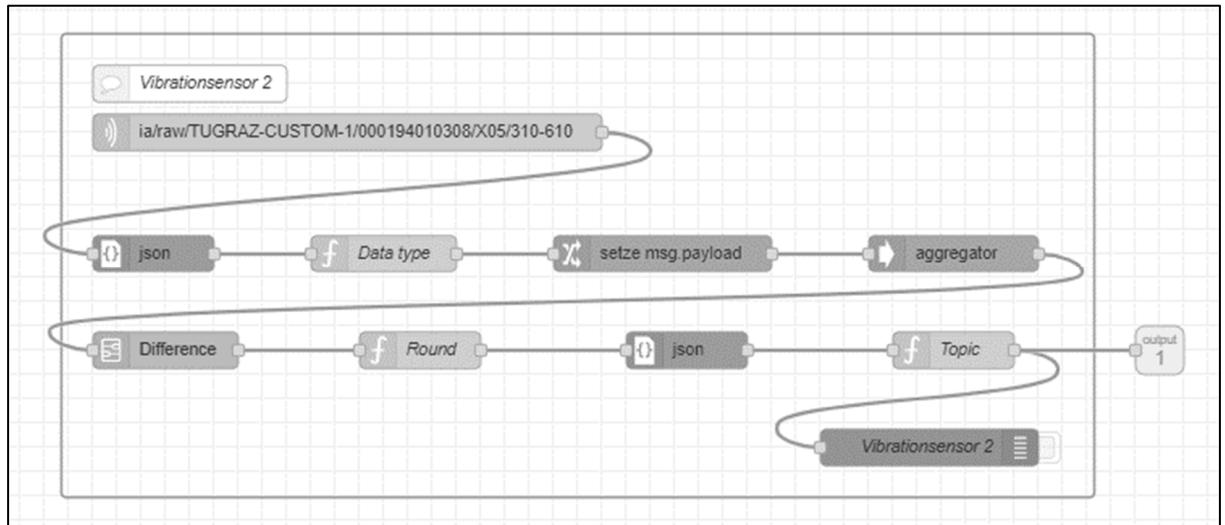


Figure 70. Data processing vibration sensor surrounding LEAD Factory

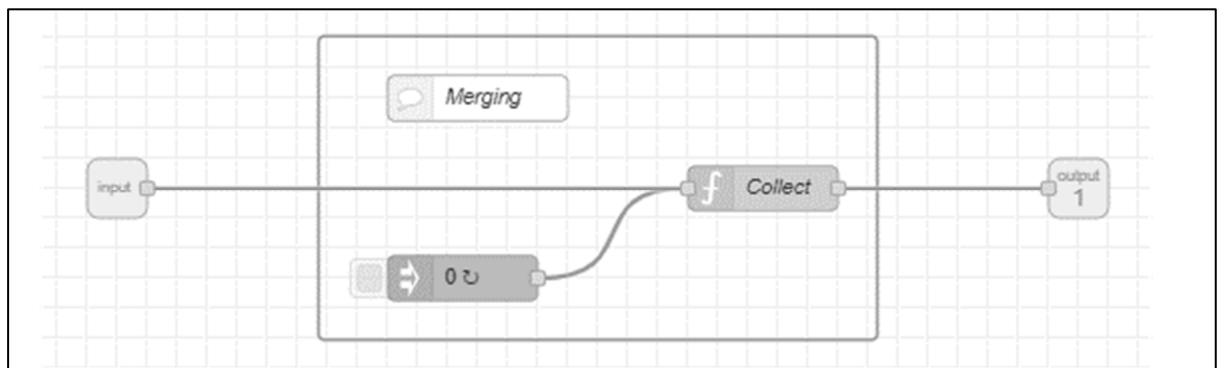


Figure 71. Merging of the processed data

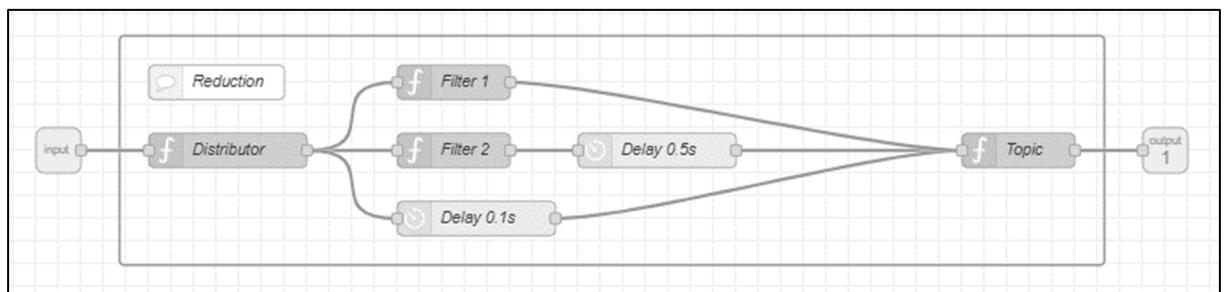


Figure 72. Downsampling of the processed data

Appendix

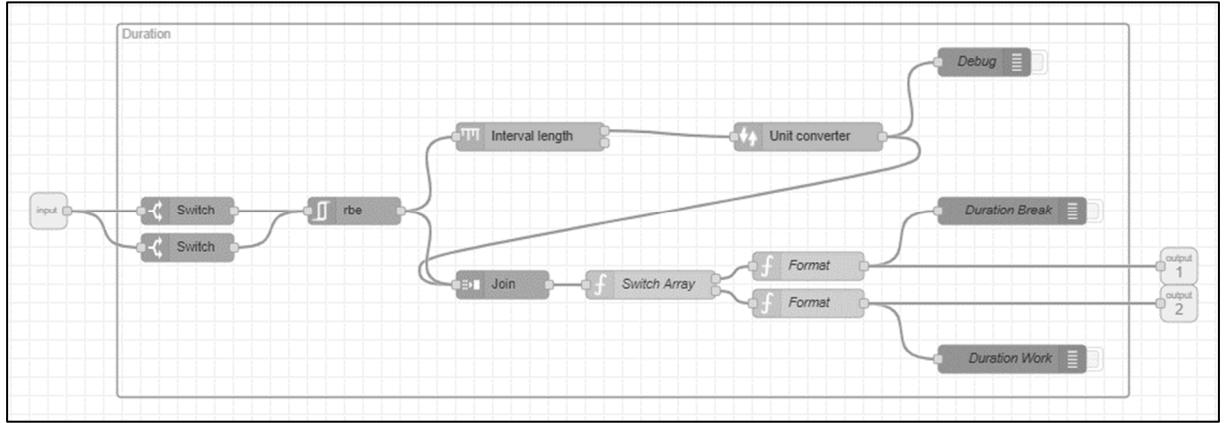


Figure 73. Processing longterm data 1

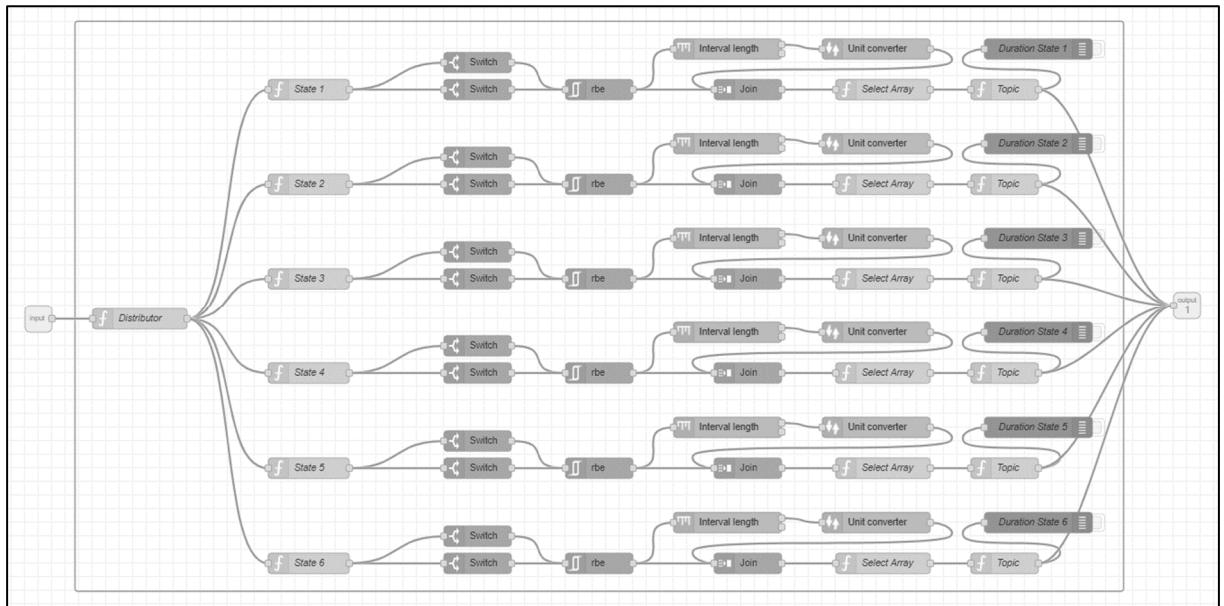


Figure 74. Processing longterm data 2

Appendix

Appendix D: Survey on the weighting of the comparison criteria

