Aalborg University
Department of Materials and Production
Manufacturing Technology
4th Semester Master's Thesis

Enabling Digital Transformation in SMEs: A Low-Cost, Modular Approach to Data Acquisition and Utilisation in Manufacturing Processes

AALBORG UNIVERSITY
STUDENT REPORT

AALBORG UNIVERSITY

STUDENT REPORT

Manufacturing Technology / Virksomhedsteknologi
Aalborg University
http://www.mp.aau.dk

Title:
Enabling Digital Transformation in SMEs: A Low-Cost, Modular Approach to Data Acquisition and Utilisation in Manufacturing Processes

Theme:
System Design

Project Period:
Feb. 2025 - May 2025

Project Group:
Group 4

Participants:
Jeppe Andersen-Otte

Supervisor:
Chen Li
Martin Bieber Jensen

Number of Pages: 59

Date of Completion: 30. May 2025

Summary

This Master's Thesis addresses some of the challenges small and medium-sized enterprises (SMEs) face when attempting to digitally transform their processes and adopt technologies of Industry 4.0. Many SMEs struggle with such transformation due to limited resources and a lack of technical know-how.
Through a combination of literature review, market analysis, and technical exploration, requirements for a general-purpose solution were outlined, covering communication protocols, hardware interfaces, and system infrastructure. An analysis of existing products on the market used for digital transformation helped identify commonalities, giving an insight into why such solutions might not be more widely adopted by SMEs. A test case was developed to simulate a scenario faced by an SME seeking to digitally transform a manufacturing process. In this case, the solution was to be implemented in a robotic screwing cell at Aalborg University, which also served as the foundation for formulating the final problem statement guiding the thesis:
"How can a system be designed to extract and utilise available data from manufacturing processes, shown through the presented screw cell case, while enabling the integration of new data sources, through ease of use, intuitive design, while being deployable on off-the-shelf consumer hardware components?"

The solution was designed to extract and utilise existing data from the manufacturing process, while also enabling the integration of additional data sources, with a focus on ease of use and deployment using off-the-shelf consumer hardware. When implemented into the case setting, it leveraged open-source software and containerised architecture, deployed on a Raspberry Pi and a mini PC, achieving data acquisition with an average latency of 45.6 ms and data visualisation, all at a cost of approximately €300. It also supported the addition of new sensors, enabling insights into processes which only have legacy PLCs with limited data output.
The results demonstrated that SMEs can begin their digital transformation journey affordably, without relying on system integrators. However, while the system lowers the technical and financial barriers, it still requires a certain level of technical knowledge of the process and devices used to configure and deploy. Further development is needed to improve usability and long-term robustness.
In conclusion, the thesis shows that digital transformation for SMEs can be made more accessible through modular, low-cost solutions built with open-source software. This work lays the foundation for the adoption of Industry 4.0 among SMEs, with future work focusing on reducing the need for technical know-how and improving scalability and robustness.

Table of Contents

Preface

This Master's thesis has been written as an independent project by a 4th-semester Master's student in Manufacturing Technology at Aalborg University, using the LaTeX.

The motivation for this thesis stems from a challenge personally encountered during industry experience: small- and medium-sized enterprises (SMEs) often face significant barriers in adopting digital technologies. This observation was later confirmed through broader research, highlighting that many SMEs struggle with digital transformation due to limited resources, lack of technical expertise, and complex system integration requirements.

The objective of this thesis is therefore to explore and propose a solution that can help reduce these barriers, making it easier for SMEs to begin their digital transformation journey and benefit from the advancements of the Fourth Industrial Revolution.

Writing this thesis has highlighted the complexity involved in developing a solution that can support the digital transformation of processes. The work provided valuable insight into the real and pressing challenges currently faced by the industry, particularly by small- and medium-sized enterprises.

Generative AI tools have been used in the preparation of this thesis for grammar and language refinement.

Nomenclature

Abbreviation	Description	Unit
AAU	Aalborg University	
SME	Small to medium enterprises	
IIoT	Industrial Internet of Things	
ESG	Environmental, Social and Governance	
SME	Small to medium enterprises	
PLC	Programmable Logic Controller	
ERP	Enterprice Resouce Planning	
MES	Manufacturing Execution System	
SCADA	Supervisory Control and Data Acquisition	
KPI	Key Performance Indicators	
CAD	Computer Aided Design	
PLM	Product Lifecycle Management	
OPC UA	Open Platform Communication Unified Architecture	
MQTT	Message Queuing Telemetry Transport	
RBE	report by exception	
MCU	Microcontroller Units	
SBC	Single Board Computers	
CPU	Computing Processor Unit	
PoE	Power over Ethernet	
USB	Universal Serial Bus	
GPIO	General Purpose Input/Output	
Pub/Sub	publish/subscribe	
RP5	Raspberry Pi 5	
JSON	JavaScript Object Notation	
QoS	Quality of Service	
SQL	Structured query language	

1. Introduction

The digitalisation of manufacturing processes is no new phenomenon. It has been around since the 1970s, when Industry 3.0 began, and the introduction of digital technologies to processes took off, moving from manual labour production lines towards fully automated production lines. Industry 3.0 has laid the groundwork for the fourth industrial revolution, through the switch from analogue to digital processes by introducing data sources giving an insight into the processes and enabling the utilisation of more advanced manufacturing technologies.
The fourth industrial revolution, also referred to as Industry 4.0, is the interconnection between machines, sensors and data systems, leveraging the data produced by the digitalisation of processes in systems, tools and technologies to further improve the processes [1].
The digitalisation of manufacturing processes, enabling the use of tools and technologies from Industry 4.0, has shown to benefit companies implementing it, in areas such as cost savings, productivity and transparency. Which can result in the success and growth of businesses, and getting an advantage over possible competitors [2].

Even though digitalisation is no new phenomenon and Industry 4.0 has been around since 2011, the undertaking of the digital transformation and its implementation is happening faster and is far more widespread throughout larger manufacturing enterprises, compared to small to medium enterprises (SMEs) [3].
This is because larger companies tend to have more resources available, such as capital, access to a global workforce and a business incentive to increase competitiveness [4]. Compared to large enterprises, SMEs do have advantages, such as they often have lower organisational complexity and fewer internal barriers to collaboration, making them more agile [4]. However, SMEs take on a larger financial burden when trying to do a large-scale digital transformation while also possibly lacking the knowledge, leading SMEs to require support from external partners [5]. Industry 4.0 and the digital transformation of companies are topics spanning over a lot of different subjects, especially within the manufacturing industry, e.g. Industrial Internet of Things (IIoT) systems, which refers to systems of sensors, instruments, and other devices interconnected with industrial computer applications. It is these IIoT systems which need to be implemented within companies to accomplish the digital transformation and be able to utilise and benefit from the different tools within Industry 4.0. IIoT is important now, but will also become more important for companies to implement in the future. Especially within the European market, where EU regulations and market demands are continuously evolving toward a greener future, leveraging IIoT

can help SMEs enhance transparency and traceability across their operations, which can help with the compliance of ESG (Environmental, Social, and Governance) and other regulations, although often targeted at larger companies, SMEs are frequently affected by them, particularly when supplying goods to larger companies [6]. As a result, SMEs must also comply with these regulations, and this will likely become even more critical as environmental concerns gain greater focus in the future.
Furthermore, the interconnectivity and insights provided by IIoT can be utilised in various ways. For example, it can improve communication and integration between suppliers and consumers across the entire supply chain, enhancing accountability and efficiency [7]. Additionally, IIoT can increase production flexibility, enabling SMEs to adapt to growing trends such as mass customisation [8].
As research suggests that there is no 'one size fits all' solution for the digital transformation of companies, highly specialised knowledge is needed to utilise and implement current products on the market, and therefore investments in technology by itself are not enough for a successful digital transformation of SMEs. It has become clear through research that for a digital transformation of SMEs to be successful, the human-centred aspects required to enable digitalisation are just as important as the technology-driven aspects [9]. If a general system were available that eliminated the need for specialised knowledge to implement and use the technologies involved in digital transformation, it could add value by removing much of the know-how previously required. This would allow SMEs to implement such systems with minimal or no assistance from external partners, while also reducing or removing barriers present that hinder SMEs digital transformation.

1.1 Barriers hindering the digital transformation of SMEs

Throughout the literature, a recurring theme is the barriers SMEs are facing, which are hindering their digital transformation, some of which have already been mentioned. Researchers have identified 10 barriers towards adopting the technologies of Industry 4.0 through the analysis of 24 Danish SMEs [10]. The barriers which are expressed most often by the companies are: lack of understanding of what Industry 4.0 is and the value of it for the company, lack of competences, and getting the organisation onboard to ensure support and commitment in the transformation process [10], which underscores that the human-centred aspect is a big part of digitally transforming companies. It highlights that before the digital transformation of companies can commence and be successful, the companies have to be enlightened and educated. But it can also be viewed as the information and technology currently available on the market isn't for the layman. The layman might not understand the subject and

terms in the information, or the utilisation and complexity of the technology and its integration into companies. SMEs might, at that point, deem that the knowledge acquisition required to take on the task of digitally transforming the company with internal resources will use up too many resources, leading to the only opportunity for SMEs to achieve a digital transformation being to hire external partners. This knowledge gap present at SMEs can also be used to see how digitally mature the companies are. It has been shown that because of the lack of knowledge within the field of digitalisation, their capability of integrating it into the company is where SMEs struggle the most, portraying a low digital maturity level of the companies [9].

Other barriers were also identified by research, such as the "lack of time or funds to support Industry 4.0 projects" [10]. It can be argued that one of the reasons SMEs might not have the funds to initiate a digital transformation, is that because they do not have the know-how about the digitally transforming processes, it can be hard for companies to determine which systems and hardware they should utilise for their digital transformation since certain knowledge for a hassle-free implementation, setup, maintenance and use is required. Especially if companies need to rely on costly external partners for supplying and integrating the system, as these can come with a higher upfront cost than if the companies had the opportunity to do it themself. Furthermore, the hardware and software currently available on the market related to digitalising the manufacturing industry have a relatively high price compared to stand-alone components such as single-board computers, microcontrollers, and open-sourced programming languages for making the software. The high price of components combined with the price needed to hire external partners can make it difficult for SMEs to begin their digital transformation and step into the era of Industry 4.0, utilising its tools and benefit from it.

Based on the technical decisions companies make throughout their operation, they risk taking on technical debt. Technical debt is a term which describes the possible negative impacts of technical decisions [11].
Technical debt can arise if companies decide not to invest in technologies that benefit their operations. This might happen if the management team decides to invest in other areas they determine to be more important, which might lead to the technology investment being postponed, because of some of the barriers mentioned earlier.
Technical debt can have adverse consequences, negatively impacting companies in different areas of their business, some of which are their performance, customer relationships, implementation possibilities of new functions, increased costs and restrictions of maintainability, which ultimately leads to financial losses [11].

For companies to get out of technical debt, they need to prioritise the management of their debt. This does not only include integrating new beneficial technologies but also managing or building up an infrastructure which can support the integration of said technologies [12].

Even though research suggests that investment in technology itself is not enough for a digital transformation is successful [9] [10], this thesis will only look at how the development of technology can reduce the size of the barriers SMEs are facing by aiming to design and develop a solution to assist their digital transformation, through digitally transforming their processes. The purpose is to reduce the integration complexity of hardware and software needed in the digitalisation of processes to reduce the knowledge gap, making it possible for companies that lack the competencies and resources, lagging behind competitors and building up technical debt, to begin their digital transformation. In today's day and age, it is not a question of whether digitalisation should be introduced, but rather how it can be introduced and how fast, so that companies can maintain or achieve a competitive advantage. Further, a goal is to reduce the initial investment needed for SMEs to begin their digital transformation by removing the need for hiring external partners through the reduction of the current knowledge gap and making it possible to utilise off-the-shelf hardware components as a host for fabricated software. This leads to the research question for the thesis:

How can a general solution based around digital infrastructure be designed to reduce the barriers SMEs face in digitally transforming their processes, by emphasising ease of use, intuitive design, and easy setup using low-cost off-the-shelf hardware components?

2. Problem analysis

As previously mentioned, Industry 4.0, digital transformation, and IIoT are broad terms covering many concepts. Since the thesis primarily focuses on developing a solution to facilitate the digital transformation of SMEs, it is important to clarify how the term 'digital transformation' is used and what it entails going forward. When referring to digital transformation, it specifically relates to the digitalisation of manufacturing processes, through implementing infrastructure that enables data extraction from existing processes through hardware such as PLCs or sensors and makes this data available to systems, such as visualisation platforms. To get an understanding of what is required to design and develop a technology-based solution which can reduce the size of the barriers currently present when talking about the digital transformation of SME processes, it is crucial to understand what is required for this transformation. Further, it is important to get an understanding of what is required of a solution for integrating seamlessly with existing and new infrastructure, while also get an overview of some of the products currently on the market to understand and determine why they are not used more widespread, and also determine how the solution will be validated to ensure that it fulfils the requirements and works as intended.
The problem analysis will first go through what is required to digitally transform a process by looking into what the purpose of the transformation is and how IIoT can assist in this. Then, different kinds of infrastructure which can be utilised to accomplish the digital transformation will be presented. A market comparison will be made to get an insight into some of the available products on the market, meant for the digitally transforming processes within the industrial sector. To validate the solution, a use case will be described, with the purpose of mimicking a real-world scenario where a process needed to be digitally transformed, which will be used to test and validate the solution. Throughout the different sections of the problem analysis, requirements will be formed based on the knowledge gained, outlining the requirement list for the general solution. Lastly, adapted case-specific requirements will be presented, which will later be used to validate the solution during testing.

2.1 The Digital transformation of Processes

The task of digitally transforming already exciting processes can vary in size and complexity depending on which technologies are already present and incorporated. This section will explore what is required to digitally transform an already existing

process, which might be classified as legacy, with little to no insight into it and no communication with systems throughout a company, to an insightful process which can communicate with other systems through IIoT, entering the era of Industry 4.0.
To get a better grasp of what kind of processes the goal is to digitally transform, the term legacy process will be described. The term "legacy" is often related to things which are outdated or carried over from an earlier time, especially used when talking about technologies or computer systems [13]. Hence, the term "legacy processes" will be used for processes which consist of and utilise outdated technologies (hardware and software) and infrastructure, which do not provide the same features, insight and capabilities as new technologies. Therefore, a legacy process will be broadly outlined as processes utilising technologies and infrastructure from the Industry 3.0 era and prior, which do not accommodate the goals of IIoT.
The goals and motivations behind digitally transforming processes through IIoT are to address the challenges inherent in traditional industrial-era methods of handling data across companies. This transformation aims to seamlessly integrate all relevant data from diverse sources throughout the entire organisation and its systems, ensuring interoperability, by implementing standardised, open, and machine-readable interfaces across the infrastructure [14].

A typical example of how the Industry 3.0 era has handled getting data from processes and utilising it in an organisation is through proprietary automation stacks [14]. This means that the technologies currently present are probably custom-tailored solutions for the specific company based on technologies from one vendor, which have limited interoperability with other technologies from other vendors, creating a barrier to the interoperability which is sought after in IIoT.
The automation stack, synonymous with the automation pyramid, consists of 6 levels representing the different levels which can be present in an industrial production [14] and is illustrated in Figure 2.1.

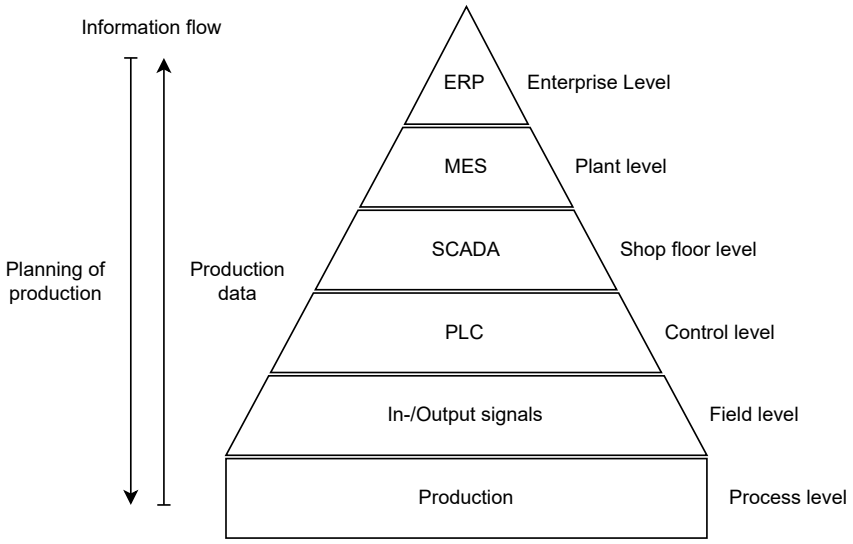


Figure 2.1: The figure illustrates the automation stack, which represents the different levels which can be present in an industrial production. Further, the information flow is illustrated through arrows, from where it originates and which direction it has through the stack. The inspiration for the figure is drawn from [14].

The process level is where the actuators and sensors are present, creating data about the processes. The field level is the interface between the production processes and the rest of the automation stack, utilising input and output signals for communication. The control level uses technologies such as PLCs (Programmable logic controllers) to control the equipment implemented in the processes. At the shop floor level, SCADA (Supervisory control and data acquisition) systems are utilised for the equipment in the processes and usually provide functionalities through HMIs (human machine interfaces) to monitor the production. At the plant level, all the production data is collected together with the planning of production in MES (Manufacturing Execution System) systems. At last, at the enterprise level, ERP (Enterprise Resource Planning) systems are used for planning operations and procurement for a company [14].
Information is flowing from the ERP system down through the automation stack regarding how the production should be planned and executed, while when the production is running information in the form of production data flow up and is condensed through the stack, and ends up as KPIs (Key Performance Indicators) in the ERP system [14]. Looking at the automation stack, an insight is given into which systems the solutions should be able to interact with to digitally transform processes by utilising IIoT. Further, it also makes it clear why it can be such a daunting task for SMEs to digitally transform their processes as there are many different interfaces between all the systems which has to be implemented individually to reach the ideas

and goals of IIoT of interoperability and seamlessly integrate all relevant data.

As mentioned the purpose of digitally transforming processes through IIoT is to overcome the problems which arise with the proprietary automation stacks by eliminating the interfaces between the different levels of the stack, focusing on a company's operation in a holistic approach instead of implementing systems only focusing on production [14], making it possible to utilise all relevant data in the entire corporation in different systems without having to make costume individually connections between the different systems, as visualised in Figure 2.2.

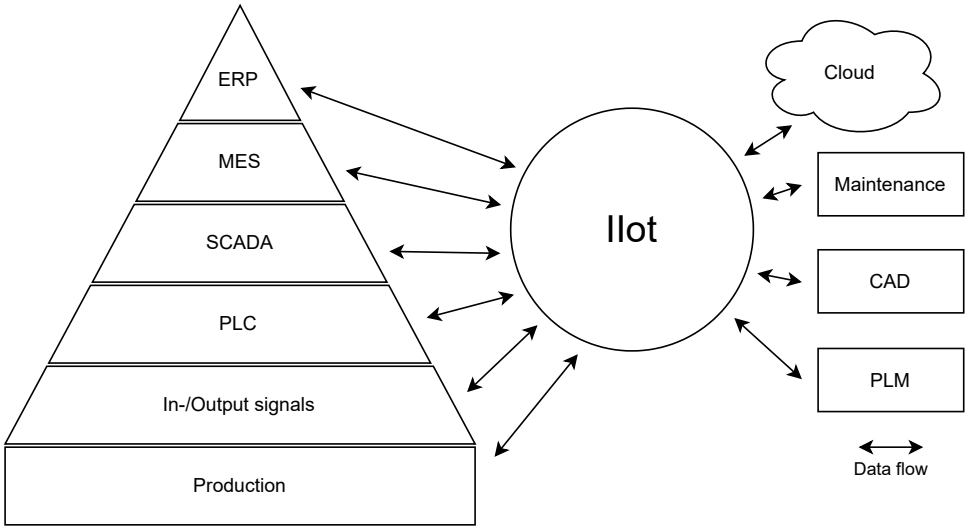


Figure 2.2: The figure shows how IIoT can be used to have a common interface between all the different systems within a company to make it easy to integrate and utilise relevant data across operations.

To digitally transform manufacturing processes through IIoT, certain digital infrastructure, therefore, has to be present, either implemented or added to already existing infrastructure to achieve standardised, open and machine-readable interfaces to ensure interoperability, because the currently implemented proprietary automation stack does not necessarily provide these interfaces.
This forms the first requirement for the solution as it has to conform with the idea behind IIoT that the interfaces must be standardised, open and machine-readable to ensure interoperability.

1. The solution must ensure interoperability by providing standardised, open and machine-readable interfaces.

2.2 Infrastructure for Digital Transformation

The infrastructure required for digital transformation may vary depending on the age of the process and the hardware and software it involves, as technological capabilities evolve over time and different vendors often use specific interfaces.
The infrastructure for a digital transformation can be implemented through two methods: invasive or non-invasive. The invasive method is where the technologies of the legacy process are "invaded" by the new technologies. This could be IIoT devices, which are connected to a PLC to pull the data from the PLC through a communication protocol by requesting specific data which are available in the PLC. Then, when the IIoT devices have received the desired data, they will act as a gateway to the implemented infrastructure and send it to the desired locations where it is needed. An example of an invasive method is illustrated in Figure 2.3.

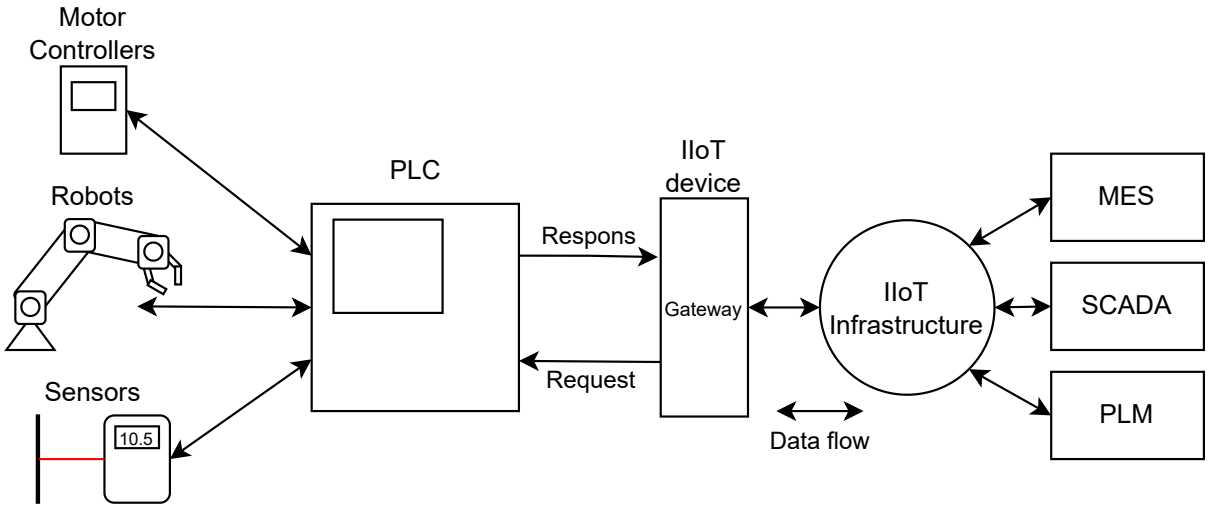


Figure 2.3: The figure illustrates how an invasive infrastructure can be implemented and request data from an already existing process and send it to desired system through the implemented infrastructure.

Depending on the state and knowledge of the process and technologies within it, the invasive method might not be suitable, as the knowledge of the workings of the program within the PLC might not be available any more and therefore it can be hard to extract any useable data, hence the non-invasive method can be utilised. The non-invasive method does not "invade" the processes and technologies, which are already in place, but instead runs in parallel with them. This requires that, instead of being connected to the PLC and utilising the built-in sensors and data which are available, new sensors have to be added to the process to produce usable data and then communicate it through a gateway to the infrastructure. An example of a non-invasive method running in parallel with existing technologies is illustrated in Figure 2.4.

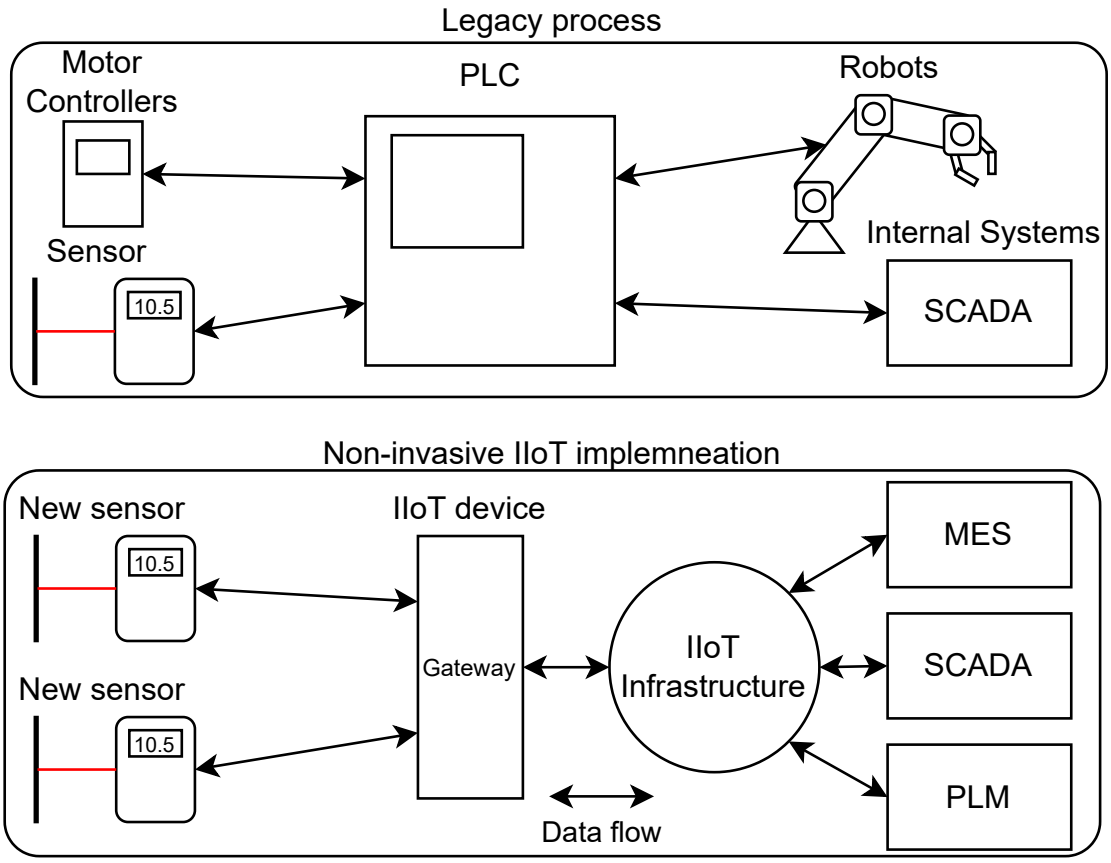


Figure 2.4: The figure illustrates how a non-invasive infrastructure can run parallel with a process already implemented in a company and send it to the desired systems through the implemented infrastructure.

For a general solution it would be beneficial if it could be used in both scenarios, where it both can connect and communicate with PLCs to extract data and send it to the places where it is needed, while also acting as a gateway for new sensors if it is not possible to get the desired data from the PLC, because of the lack of availability. Thereby, the following requirements for the solution can be formulated.

2. The solution must be able to be implemented both invasively and non-invasively into processes.
3. The solution must be able to provide the infrastructure making it possible to extract and send the relevant data to the right locations.

For the solution to fulfil these requirements, it must build up an infrastructure composed of software which can run the right communication protocols needed to communicate with technologies already present in the processes, to extract data, while also working as a gateway to send the data to the right systems. Further, the infrastructure should be based on hardware which can run this software while also

allowing for new sensors to be added if needed through dedicated physical interfaces. The next sections will look into different areas for an operating infrastructure, which are communication protocols, hardware, and physical interfaces, to determine what might need to be implemented in the solution to build up this infrastructure needed for digitally transforming a process.

2.2.1 Communication protocols in the industry

For the various devices, sensors and instruments present in the industry to communicate with each other, a communication protocol often has to be in place so that the right data can be transferred to the right location when it is needed or produced. This section will look into some of the communication protocols used within the industry, their advantages and disadvantages.

Modbus
Modbus was originally developed in 1979, and in efforts to bring it into the 21st century, a newer version of Modbus has been created called Modbus TCP/IP making it possible for Modbus to communicate over Ethernet, making systems with Modbus able to connect and communicate with infrastructure and systems using the same widespread networking standard [15]. Modbus is based on a client/server architecture and, at its core, works as a Request-Response model, where the client requests data from a server and then waits for the response, as illustrated in Figure 2.5 and can be used to communicate between devices and sensors [15].

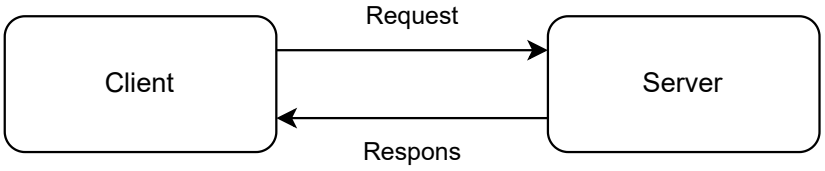


Figure 2.5: The figure shows how the client/server architecture works, by having a client which requests data from a server and then the server returns with a response.

Some of the advantages of Modbus are that it is lightweight and has well-defined functions, making it simple and efficient to implement. Some of the disadvantages are that it is based on a request-response model, which means that unnecessary requests are performed if the data hasn't changed, and its ability to scale as the performance can degrade the more devices connected and broadcasting messages [16].

OPC UA
OPC UA (Open Platform Communication Unified Architecture) covers different communication architectures but is primarily implemented as a client/server architecture,

in the same way as Modbus, illustrated in Figure 2.5. With over 1200 pages of documentation, the protocol has a wide range of functions and specifications that can be used within the automation industry [17, 18, 19]. Some of the advantages of OPC UA are that it has a lot of functions and specifications, making it possible to connect to a wide range of applications while also being a secure protocol. The wide range of functions and specifications is also one of the disadvantages because the protocol is so large and has so many features that it can be hard and complex to implement, and it also has a large bandwidth footprint compared to other protocols.

MQTT & Sparkplug B

MQTT (Message Queuing Telemetry Transport) is a simple, lightweight communication protocol and is designed based on the principle to minimise network bandwidth and device resource requirements. The protocol is based on a publish/subscribe architecture and model, where devices can publish categorised messages to a topic within a broker, and subscribers of these topics will then receive the messages [20], which is illustrated in Figure 2.6.

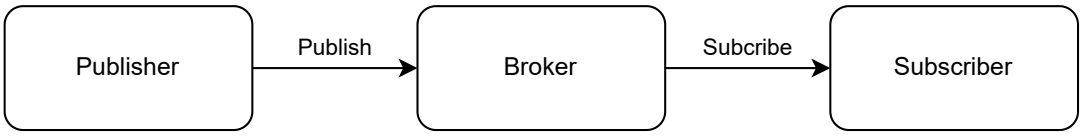


Figure 2.6: The figure shows the publish/subscribe architecture, where the publisher sends data to a broker, which distributes it to subscribers.

One feature which makes MQTT desirable is the feature to report by exception (RBE), which is a method of communication which only sends messages when a value has changed. Further, MQTT only consists of 80 pages of documentation and was intentionally designed not to standardise some of its infrastructure to provide maximum flexibility across different sectors, which might choose MQTT for their infrastructure [18, 21]. However, for MQTT solutions in specific sectors to be interoperable, infrastructures such as the Topic Namespace, Payload representation and session state must be standardised [21]. This disadvantage of having too much flexibility can be helped with additional protocols build on top of MQTT such as Sparkplug B, whose purpose is to define a standard structure for MQTT, so that the data from various sources can be easily understood and used by different applications, making it a good candidate to use as a messaging protocol within the sector to ensure the interoperability.
It should be mentioned that these communication protocols mentioned in the thesis are just some of the many protocols which are utilised within the industry. Other but not all industry communication protocols are: CC-link, Ethernet/IP, EtherCAT, Profibus, Snap 7 and Profinet.

Some of the protocols investigated have been on the market for a long time and are highly integrated into the industry, while others are newer and are less integrated. For a general solution which can connect to different technologies, which might vary in when they were produced, the number of communication protocols the solution has to accommodate might increase, as some older technologies might not have the capabilities of using newer communication protocols. Some communication protocols might be better in the overall infrastructure because it is less complex, but in other places of the solution, a communication protocol might need to be implemented before it can communicate with certain technologies. Therefore, a solution can be said to have two kinds of protocols categorised into two areas: external-bound protocols and internal-bound protocols. The external-bound protocols are the ones which the solution has to utilise as an interface for communication with already existing technologies and devices within processes. The internal-bound protocols are the ones used within the infrastructure of the solution to communicate with different systems of the infrastructure. These two categorisations are used to form requirements for the solution of which protocols to use where, and are the following:

4. The solution must be able to communicate with already existing technologies and systems through a range of different external-bound communication protocols.
5. The infrastructure of the solutions must have an internal-bound communication protocol, which is standardised and has open interfaces.

2.2.2 IIoT Gateways

IIoT gateways are, in general, computing devices positioned on the edge and are used within the digital transformation to link industrial processes and digital infrastructure, allowing seamless communication between them. The IIoT gateway is in charge of collecting data from the different devices and sending it to the right location where it can be utilised for different purposes. Depending on the capabilities of the computing device used as the IIoT gateway, the number of functionalities of the gateway will differ. If it is desired to move some of the computing tasks from, for example, the cloud, such as filtering, processing and analysing data to the edge for quick response and lower the bandwidth usage within the digital infrastructure, and only sending the most relevant data to the cloud, more computing power of the IIoT gateway is needed compared to if the only task is the capture and send data. Some of the common hardware components in use for IIoT gateways are microcontroller units (MCU), single board computers (SBC) and Industrial personal computers (IPC).

Micro controller Units
An MCU is an integrated circuit, often on a small circuit board with processors, memory and peripherals, designed to do specific tasks within a system without requiring an operating system. MCUs are often cost-effective stand-alone units, making it possible to process signals, control devices, or work as an interface between sensors at a relatively low price compared to other hardware.

Single board computers
An SBC is a complete computer system integrated on a single circuit board, from its computing processor unit (CPU), memory, input and output devices, and other features which are required in a computer. They are often used as embedded computer controllers within the field of machine control and automation. As everything needed for the SBC to function is on a single board, the price can be kept low by reducing the number of parts connections compared to what would be used in a standard computer.

Industrial personal computers
An IPC is simply a computer intended for industrial use. They differ from normal computers by following higher standards regarding dependability and precision, making them generally more expensive. They have a wide range of use cases, such as controlling processes and acquiring data. The lines between IPC and SBC can be blurry as an SBC can be an IPC if it lives up to standards and requirements, but an IPC does not necessarily have to be an SBC, as it can contain multiple boards or parts to add peripherals.

Physical interfaces
When connecting devices for communication and transferring data, and a device does not have the capability of wirelessly communicating, physical interfaces are needed. Some common physical interfaces present on industrial devices are: Serial port, RJ45, Universal Serial Bus and General Purpose Input/Output.
The Serial Port is the port used to communicate with devices which are utilising serial communication protocols such as RS-232.
The RJ45 interface is a common interface which can be used with numerous Ethernet communication protocols, such as OPC UA and Modbus TCP. Some RJ45 interfaces can also be utilised for the feature called power over Ethernet (PoE), which also sends power through the cable while being able to communicate with the connected device.
The Universal Serial Bus (USB) interface is a widespread interface and a standard interface for many devices. It was made to replace the older serial port as the need for higher bandwidth, which the serial port could not provide, increased.
The General Purpose Input/Output (GPIO) interface is often used for devices which

do not have some of the other interfaces mentioned. The GPIO, where the wires of the devices are connected directly to the pins, can be controlled by software and send or receive signals.

These four different physical interfaces are just some of the physical interfaces which are used within industries to communicate and transfer data between devices. An illustration of the different physical interfaces can be seen in Figure 2.7.

Figure 2.7: The figure depicts 4 common physical interfaces which are used within the industry.

With the knowledge of common hardware used for IIoT gateway devices and common physical interfaces, it is possible to further investigate what hardware should be utilised in the solution to be general while also being cheap, which assists in reducing the barriers present in the digital transformation. This generates the following requirements for the solution:

6. The solution must be based on commonly used hardware.
7. The hardware utilised for the solution must have common industry interfaces.

2.3 Market Comparison

A comparison will be made of selected hardware and software products currently available on the market that are used to digitally transform processes through IIoT. The purpose is to get three outcomes: From the hardware comparison, the outcome will be an indication of what features products on the market already provide at given price points, which can be used to determine what hardware to utilise for the solution while also indicating what a possible price range of the solution will be. From the software comparison, the outcome will be knowledge about transparency in the market regarding the pricing of products, while also understanding what the upfront and monthly costs of software are. The third outcome will be a price estimate of what it will cost a company to begin a digital transformation by utilising the compared hardware and software and applying it to a process line. The products which will be analysed will only be products and services which have publicly available pricing.

2.3.1 Market Comparison: Hardware

As mentioned in Section 2.1, to digitally transform already-present processes, a gateway may be necessary to be in place to extract the available data from the process and send it to the infrastructure. There are many different edge devices which can be used as a gateway in the market, and also at vastly different price points, closely related to their capabilities. The hardware which will be looked at is some of the devices which were classified as usable for gateways in Section 2.1, i.e Micro controller units (MCU), single board computers (SBC), and industrial PC's (IPC). These components differ in capabilities, structure, and performance, each offering unique advantages and disadvantages. While they cannot be directly compared on a one-to-one basis, they will be evaluated against one another to provide an overview of what is available on the market. This comparison will highlight the features consumers can expect and the trade-offs involved in choosing specific components of each type across different price points. The comparison can be seen in Table 2.1.

Type	Product Examples	Price Range	Advantages	Disadvantages
MCU	STM32, ESP32, Raspberry Pi Pico	5 € - >100 € [22]	Small scale, Cheap, Power efficient, Available	Limited Processing power, Limited Memory, Development Complexity
SBC	Raspberry Pi, Jetson Nano, Asus Tinker	21.5 € - >400 € [23]	Affordable, Easy to deploy, Available, Versatile	Often fixed configuration, Often limited expandability, Can struggle performance tasks
IPC	Siemens, BeckHoff, Onlogic	175 € [24] - >10000 € [25]	High Performance, Reliable, Scalability, Connectivity	Higher Cost, Typically larger, Overkill for simple tasks

Table 2.1: The table shows a comparison between MCU, SBC and IPC, to showcase their price points and together with advantages and disadvantages

To keep the comparison brief, it should be noted that the examples of products mentioned in the table are only a fraction of what is available on the market. This makes it clear that it is not the lack of choices which is the reason for the lack of digital transformation, however, it can be a cause of confusion for SMEs. If the SMEs do not have any prior knowledge within the field, it can be very overwhelming with all the different choices which are available on the market regarding hardware, and what to choose to ensure that the specification is correct for their specific use case.
The price range which has been noted is based on prices available at consumer suppliers and is a picture of how the prices are at two suppliers and could differ depending on where in the world it is bought and in what amount. The question about whether looking at a consumer-supplier for some of the components gives an accurate view of what it will cost for SMEs to buy hardware for their digitalisation is deemed to be okay. This is because the purpose of the solution is to be able to utilise off-the-shelf hardware available to everyone.

2.3.2 Market Comparison: Software

To make a comparison of available software products on the market, a set of rules is set up, which the products have to follow before they will be considered in the comparison. These rules are based on the mindset of transparency and clearness of how much it will cost to digitally transform processes while also getting tools which can utilise the data from the processes. Therefore, the rules will be as follows: 1. The software products must provide all parts of the infrastructure, from data extraction, communication routing and data utilisation tools. 2. The software must be able to integrate with already existing technologies present in processes. 3. The price of the software must be publicly available and concise in what it covers. 4. As support might be needed for less experienced companies, support must be included.
While looking at the market, a common observation made was that companies providing software-related products to IIoT do not publish the prices of their products or services, which might be because they do not want to publish their prices to their competitors. However, the missing public price can also be a factor that explains why SMEs still lag behind larger companies. They will have to contact the provider, wait for a reply and have a conversation with a sales representative before they can get an estimated cost for the infrastructure needed to digitally transform their processes. This can make the process more confusing and less transparent, hence, the SMEs might feel overwhelmed and not take the initiative to write to the providers to begin their digital transformation, and instead wait for a more transparent and concise product or for someone to contact them with an offer. The companies which provide the software for this comparison are the following: Inductive Automation, Tulip, Fact Bird, and Machinemetrics.

In the effort of making a fair comparison of the different suppliers, the features of the product have to match as much as possible. Therefore, for some products, extra packages had to be added to provide the features which were missing from the basic product. The prices are based on a setup containing 10 devices, as some companies set this as a minimum number of devices. The comparison can be seen in Table 2.2.

Company	Product	Packages	Upfront Cost	Mothly Pricing
Inductive automation	Ignition	Ignition plateform, Perspective Module, Edge IIoT, Basic care	14,160 €	163 €
Tulip	Essentials subscription	Included packages at website	920 €	920 €
Fact Bird	Single Site subscription	System base fee, Production insight	2,825 €	2,825 €
Machinemetrics	Professional subscription	Included packages at website	1,385 €	1,385 €

Table 2.2: The table shows a price comparison between 4 different IIoT software providers. The prices are based on their different platforms and on connecting 10 devices/machines, as this is the lower limit for some providers.

In the comparison there are 2 prices present: An upfront cost and a monthly payment. This is because most of the company's products are based on monthly subscription services. What the comparison highlights is that Ignition is not based on a monthly subscription service in the same way. On the other hand, there is a much higher upfront cost because you pay for the modules once and then the monthly payment is only for support and updates of the programs, which was a requirement for the products to be used in the comparison.
It should be noted that the comparison is only looking at the prices of the products and is lacking some insight in areas such as performance, how easy they are to implement, their quality of customer support, interoperability and scalability. These things have not been looked into, as it would require an extensive analysis through using the products and services before it could be determined. The prices of the products still give an insight into the amount of resources which a company has to invest to begin its digital transformation, which was also the purpose of making this comparison.
Further, it should be mentioned that there are a lot of products on the market which were not included in the comparison because of their lack of public prices, or their prices being incalculable without further specifications of what is needed, such as for Amazon and Microsoft IIoT platforms, as they determine their prices on multiple factors such as connection time to the cloud, messages sent, devices registered, and other parameters. It would not be surprising if SMEs also might get confused by this type of payment model.

2.3.3 Estimate of digitally transforming a process

By looking at 3 different categories of hardware and 4 different software suppliers within the IIoT sector, it is clear that what is currently on the market can become expensive and hard to navigate because of the many choices. Assuming a company would buy hardware and software at the above-mentioned price points for 10 devices as a minimum. It can cost companies from 970 € up to far above 15,000 € just in upfront cost for the hardware and software, which does not account for the time it takes to decide on a system and integrate it nor the cost related to if companies need external partners for the integration as these can be hard to estimate.
Even though the upfront cost of €970 may not seem significant, it is often tied to long-term contract commitments. For some SMEs, especially those unsure of the immediate benefits of digital transformation, this can appear to be a substantial investment and may be perceived as a waste of resources. Therefore, it is also crucial to make a solution which does not require a high upfront investment in things such as hardware or software. This can be done by making a solution which is based on open-source software and is not hardware-bound in the sense of being able to run on MCUs, SBCs, and IPCs, or by making it able to run on the premise so that no cloud providers are needed. It is evident that the remarks made in the comparison are influencing the digital transformation barriers for SMEs negatively, by making the process of the transformation expensive and less transparent, which makes some clear points which can be formed as requirements for the solution.

8. The solution must be integrable and usable without the need for contacting external partners.
9. The solution must be based on open-source software.
10. The solution must be based on off-the-shelf consumer hardware.
11. The solution must be able to run within the premises of a company.

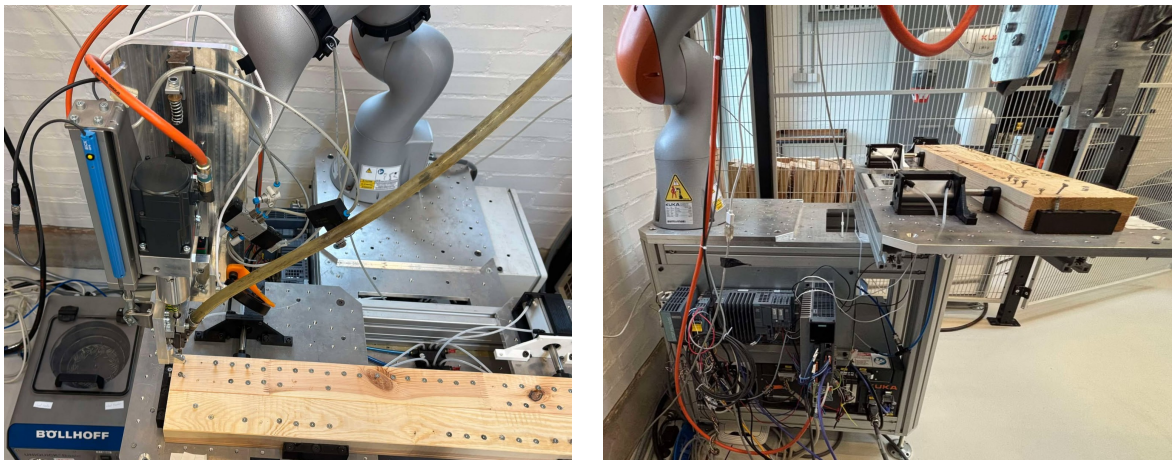
2.4 Case: Digital transformation of the screw cell

As it is important to test and validate the solution to confirm that it works as intended and to discover potential flaws and areas for improvement a case is set up for this purpose. Since the purpose of the solution is to help SMEs with digital transformation the case will mimic a setup which could be present in a process line within an SME. As the thesis is not made in collaboration with any SMEs and is solely made within the borders of Aalborg University (AAU), the process is one which already is present in the workshop of AAU. Even though the case is not directly related to an SME the

case is deemed sufficient to prove if the solution can be utilised with a process line of an SME as the components which will be in focus within the process are general components used within the industry such as a PLC and arbitrary sensors.

2.4.1 Scope of The Case

The case will take basis in the screw cell present in the laboratory at AAU. The purpose of the screw cell is to get a robot arm with a mounted screwdriver, to automatically screw screws into a board and determine if the screws have been fastened correctly. Pictures of the screw cell can be seen in Figure 2.8.



(a) Front view of the cell

(b) Side view of the cell

Figure 2.8: Pictures of the screw cell which will be used to test an validate the solution.

To get an overview of how the setup of the screw cell is and what technologies are already present in it, it is further described in the following section. This will create the setting in which the solution has to fit and what kind of technologies the solution has to be able to communicate with to fulfil the case.

2.4.2 Setup of the screw cell

The setup of the screw cell is illustrated in Figure 2.9. The setup is centred around a table where a robotic arm (KUKA LBR iiwa 7 R800) is mounted. The end effector of the robotic arm is an electric screwdriver with a screw bit matching the designated screw head. The screws are fed into the screwdriver by a tube which is connected vibration bowl screw feeder, which orients the screws, to ensure the right orientation when fed to the screwdriver. The screwdriver is controlled by a Siemens PLC (SIMATIC ET 200SP) and servo drive (SINAMICS S210). To ensure that there is contact between the screw bit and screw head, and to push the screw into the wood

while screwing, the screwdriver is mounted to a pneumatic cylinder, which moves the screwdriver when activated.

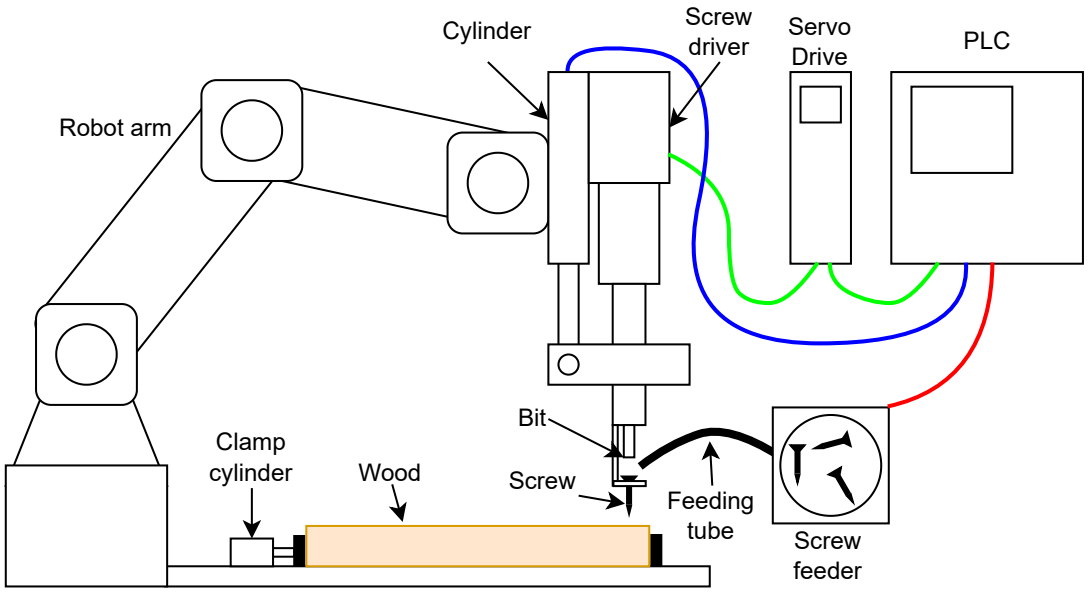


Figure 2.9: The figure illustrates the different components present in the cell. The connection between the PLC and the components it controls is also illustrated through coloured wires.

On the same table as the robotic arm, a fixture is mounted where pieces of wood can be put into. When the fixture is activated, it clamps the wood using pneumatic cylinders to ensure it is fixed in the right position. The clamping mechanism is controlled through a pneumatic valve controlled by the PLC controlling the screwdriver.

Assuming this case represents a real-world scenario for an arbitrary SME, where a PLC is integrated into a process and contains internal data, and additional sensors are mounted to generate further data. A challenge could be that this data is neither readily accessible nor available in a format that can be easily collected and utilised by other systems. The scope of this case will therefore be to extract the currently available data within the cell, transmit it to a centralised location, and enable its utilisation and visualisation to provide an overview of the process. Furthermore, the solution should be designed to accommodate future expansion, such as the integration of additional sensors. This case will be divided into three phases: Phase 1: Setup and configuration of the solution. Phase 2: Data utilisation. Phase 3: Integration of additional sensors.

2.4.3 Final problem statement

Through problem analysis, a clearer picture of the purpose of digital transformation has been established. Additionally, by examining the necessary infrastructure and comparing available market solutions, an insight has been gained into why a digital transformation can be challenging for SMEs. The presented case, which is used to simulate an SME's process, aiming to digitally transform and leverage Industry 4.0 by extracting available data and utilising it, enables the formulation of a final problem statement for the solution:

How can a system be designed to extract and utilise available data from manufacturing processes, shown through the presented screw cell case, while enabling the integration of new data sources, through ease of use, intuitive design, while being deployable on off-the-shelf consumer hardware components?

2.5 Requirements

As the requirements outlined in this report are general requirements, and as such, are not necessarily measurable, they can be quite broad and complex to define what specifically is required from the solution for them to be fulfilled. To reduce the scope of the thesis while also refining these requirements into more concrete requirements, they will be mapped onto the presented case. This case, which mimics a possible SME scenario, enables a practical evaluation of how well the proposed solution addresses the core objective of reducing the barriers that SMEs face in their digital transformation journey. To structure the evaluation, the requirements are divided into three sections that reflect the case's progression: Configuration and deployment requirements, Utilisation requirements and Extensibility requirements. The requirements will be assessed in two ways: True or False statements will be used for binary features, i.e. whether the solution includes them or not. Numerical measurements will be used for requirements where the outcome may vary from the desired value. The engineering requirements are tabulated in Table 2.3. Together with the numerical values of the requirements, the desired direction of the requirements is indicated with an arrow that will be used to specify whether the number is desired to be higher or lower. If more information about the requirements and how they have been derived is desired, an enumerated list with a longer and more in-depth description of the requirements can be found in Appendix A.

No.	Requirement	Value	Direction
Configuration and Deployment			
1.	The solution should provide at least 1 standardised interfaces used for internal communication.	1	↑
2.	The internal communication protocol must be standardised and open interfaced to ensure interoperability.	T/F	-
3.	The solution should be able to be deployed on at least 2 of the 3 mentioned hardware (e.g. MCU, SBC or IPC).	2	↑
4.	To remove the need for relying on external partners, it must be able to be deployable without.	T/F	-
5.	To reduce the resource barrier, the solution must only use free and open source software components.	T/F	-
6.	The solution must be able to be deployed on off-the-shelf consumer hardware, to ensure component availability.	T/F	-
7.	To further reduce the resource barrier, the solution must aim to minimise the cost of deployment.	T/F	↓
8.	The solution must be able to be deployed on-premises, to remove the need for external cloud providers.	T/F	-
Utilisation			
9.	To ensure data availability and real-time insight, the extraction and transmission of data should not take longer than 100 ms.	100 ms	↓
10.	To get the data from the already existing process, the solution has to act as a gateway which can extract data.	T/F	-
11.	To be able to utilise the data, the solution has to distribute the data to a visualisation component.	T/F	-
Exstensibility			
12.	As there might be a need for adding extra sensors to the setup of the case, the solution must be able to support this.	T/F	-
13.	For the solution to be versatile in an industrial setting, it must support at least 2 industrial communication protocols.	2	↑
14.	For the solution to be more general, it should have at least 3 common industry physical interfaces (e.g. RJ-45, USB, GPIO).	3	↑
15.	The solution must have at least 2 standardised interfaces used for data collection.	2	↑

Table 2.3: The requirements for the solution, as tabulated, are derived by transforming the general requirements from the problem analysis into more case-specific and tangible engineering requirements. These requirements will be assessed either as true or false, or by a specified numerical measurement.

3. Concept Design

An ideal general solution must comply with all requirements which could be present within an industry setting, while offering flexibility through a broad range of features while supporting industry standards and interfaces. It should be vendor- and hardware-neutral, ensuring interoperability with other technologies.
Such a solution could be deployed on different devices, such as MCUs, SBCs, and IPCs, simply by installing it, enabling out-of-the-box functionality. It would autonomously determine its operational context, including connected devices, communication methods, data extraction, and storage, effectively acting as a self-aware system. Its flexibility would stem from universal protocol support, eliminating manual configuration. The only human interaction required would be installation, defining which data to visualise and in which format the data is visualised. As such, an ideal solution is quite a cumbersome task to produce, this chapter will go through how a concept of a less feature-rich but still usable solution can be designed to solve the presented case while meeting the requirements.

The concept design chapter will go through how the design of the solution has been deduced. First, a functional decomposition will be made of the case to determine which essential functions the solution has to have to solve the presented case. These essential functions will be used as a pointer to determine which components to utilise within the solution. The functions for the solution will be divided into two subcategories: The backend, describing how all the data will be stored, distributed and utilised through visualisation, and the edge, describing how the data will be extracted from different devices and sent to the backend. For the solution to achieve being easy to use, it will be discussed how it is incorporated into the solution a way to manage the infrastructure and implement the principles of easy to set up and intuitive design, so that the solution can adhere to the goal of reducing the barriers present for SMEs when wanting to digitally transform their processes. A holistic view of the concept of the solution is created and depicted in Figure 3.1 to get an overview of how the data flows throughout the solution while also getting an insight into which components are a part of the solution, and how they are divided between being a part of the edge and backend.

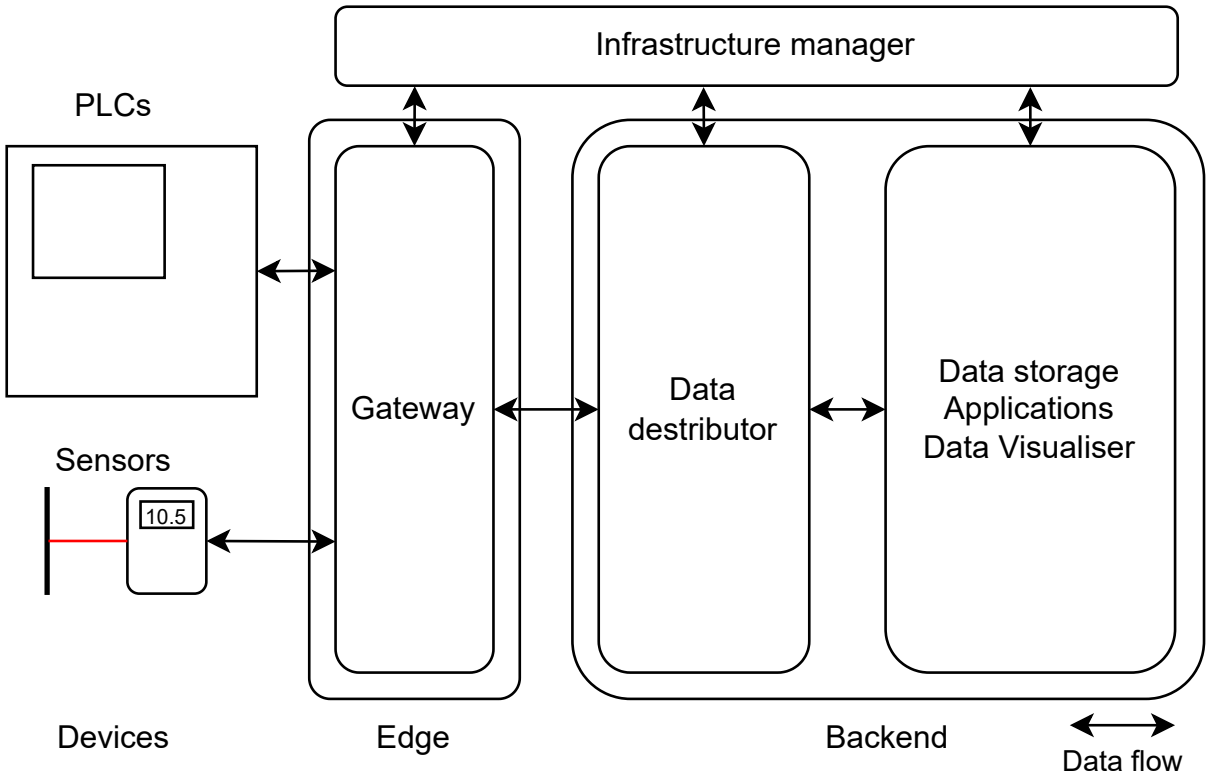


Figure 3.1: The illustration shows an overview of how the solution will be structured.

Edge: On the Edge, a gateway will be implemented, which will act as a gateway between the devices of the process, where data is produced, and the backend infrastructure. The gateway will communicate with existing devices within the process to extract relevant data, contextualise it, and forward it to the backend using the internal communication protocol.

Backend: When data arrives in the backend, it will be distributed to the appropriate destinations. The Data Distributor is responsible for ensuring that the correct data reaches the right locations in a timely manner. All relevant data generated at the edge or within the backend will be stored. Applications can be a part of the backend if specific features are desired, such as specific calculations, scheduling or notification. It is also the backend, which is responsible for hosting the data visualiser.

Infrastructure manager: To manage the infrastructure, establish connections between the process devices and the edge, and ensure seamless communication between the edge and the backend, an Infrastructure Manager will be a part of the solution. This Infrastructure Manager will serve as the interface between the user and the entire solution. Through this interface, the user will be able to set up and configure the solution.

3.1 Functional Decomposition

A shortened concise repetition of the scope of the case which has to be solved is: Extract and utilise available data from the screw cell while allowing the addition of new data sources. This is what the overall function of the solution has to be to solve the case, and will work as an outline for the solution and which components to utilise. To get a better insight into which essential functions the solution has to be able to solve, a functional decomposition of the scope of the case is made. This will make it possible to find components for the solution which can solve the essential functions, and then incorporate them into the solution. The functional decomposition is illustrated in Figure 3.2. The essential functions are grouped into which parts of the solution they belong, i.e. the functionalities which the edge or backend has to be able to do to address the scope of the case. In the illustration, there are three groups: edge, backend and edge/backend. The last group is functionalities, which apply to both the edge and the backend.

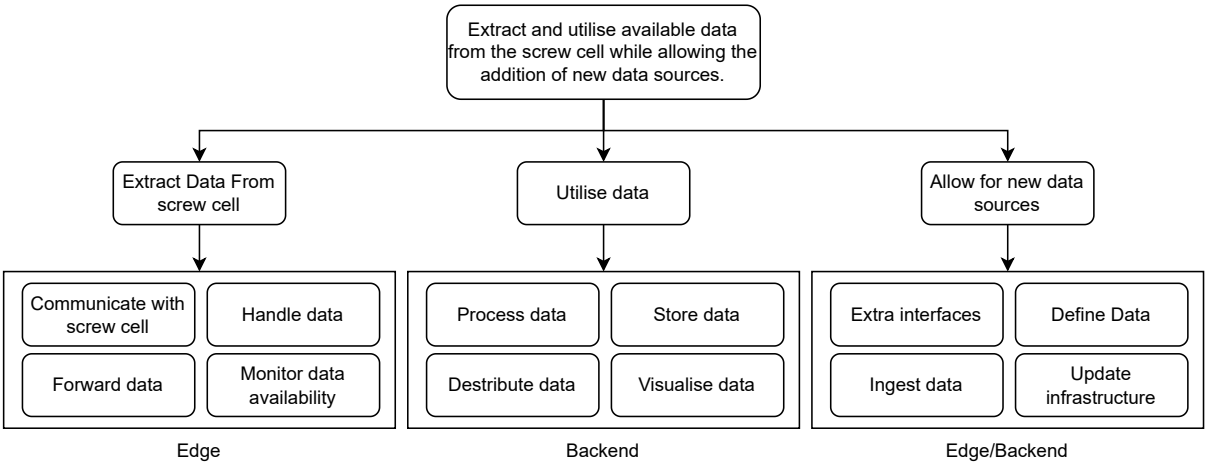


Figure 3.2: The figure shows the functional decomposition of the overall function of the solution, decomposed into essential functions. The essential functions are grouped into their respective categories of edge or backend, where they are applicable. The edge/backend category is applicable for both.

3.2 Conceptual Infrastructure Design

The conceptual infrastructure design will map out how the infrastructure of the solution can be, by combining possible components which can provide the essential functions needed for the solution, which was deduced from the function decomposition. The concept design will be described in three sections: First, the hardware components which will be considered for the solution, second, the software components, and lastly, the infrastructure manager components will be explained.

3.2.1 Hardware components

When deciding which hardware components the solution should consist of, there is an overwhelming number of options, as already mentioned in the market comparison in Section 2.3. To determine which categories of components and potential candidates within these categories are suitable for the solution, a brief discussion will be provided on possible additional requirements, beyond those already defined for the solution, that may apply to the selected hardware. This will provide the categories of components within which the solution has to be, and some possible candidates for both the edge and backend hardware components.

Edge hardware components

For the hardware that must be deployed at the edge, the requirements can vary depending on the industry and specific setting in which they are deployed. In the case of this thesis, one important factor to consider when selecting hardware components is their form factor. It would not make sense to deploy a large computer next to a cell, where space is quite limited, nor to rely on a centralised device located some distance away, as this could lead to high cabling costs, complicated cable management, and tedious installation efforts. Therefore, to bring computing closer to the edge, right where the data is produced, the small form factor and capabilities of SBCs seem to be a good fit. Some possible candidates for the edge hardware include the Raspberry Pi, Jetson Nano, and Asus Tinker Board.

The reason SBCs are a promising choice for deployment at the edge is, as mentioned, their small form factor combined with good computing capabilities, all at a relatively competitive price compared to other hardware options, as discussed in Section 2.3. Another feature SBCs often offer, compared to systems such as servers or traditional computers, is the availability of a GPIO interface, which is commonly used for connecting sensors. Further, the small form factor also makes SBCs easier to adapt for dusty, dirty, or humid environments by equipping them with specialised casings to make them dust- and splash-proof, although this feature is less relevant in this particular case, since the deployment is within a controlled environment.

Backend hardware components

For the hardware used to host the backend part of the solution, the requirements are not necessarily the same as for the edge devices. The backend is typically placed in a more controlled environment, where space constraints are less of a concern. However, it must still be reliable, available, and capable of running all backend components. For the specific solution developed in this thesis, cost-effectiveness is also an important requirement. Good candidates for backend hardware could include equipment companies already have available from previous hardware upgrades, such as mini

PCs, old laptops, or workstations, which can be repurposed to host the backend components. If a company already has such hardware on hand, it makes them an ideal candidate, as they would not need to invest in new hardware to test the digital transformation of their processes.
Since the solution proposed in this thesis is not intended for a full-scale rollout across an entire production facility, but rather as a proof of concept deployable in confined areas, such as a single cell or process line, which can help incentivise SMEs to invest in digital transformation by demonstrating its benefits on a smaller scale. Therefore, although areas such as system redundancy and cybersecurity are important considerations, the use of mini PCs or old computers is still considered a viable option, provided the solution remains confined within the company's premises and SMEs do not become reliant on it for critical operations until redundancy and cybersecurity measures are properly addressed.

3.2.2 Software structure and components

How the software is structured and which components are included will form the backbone of the solution and dictate its functionalities. The software structure of the solution will be based on a containerised setup, where each container will represent a component. A containerised structure makes it possible to package each software component and its dependencies into its own confined unit, called a container. This confinement makes it possible for the software component to be portable and to run consistently in any environment capable of running containers. For the solution, each component will perform defined tasks while being able to communicate with the other components in the solution.

Edge software structure and components
The software components, which will be deployed to the edge and function as a gateway, are designed to be a platform for interoperability between existing devices and the backend. The components will be divided into four sets of components: device components, core components, application components and a messaging bus. As mentioned these components will be based on a containerised structure, and therefore work as building blocks which introduce modularity into the solution. This is allowed through the message bus, which will provide standardised interfaces for communication between the different components. The message bus will only handle the communication between the software components of the gateway, and communication with the process devices, backend and infrastructure manager will be handled by other components. The structure of the edge is depicted in Figure 3.3. It should be noted that this architecture is inspired by the open-source project called EdgeX

Foundry, which has a similar architecture [26].

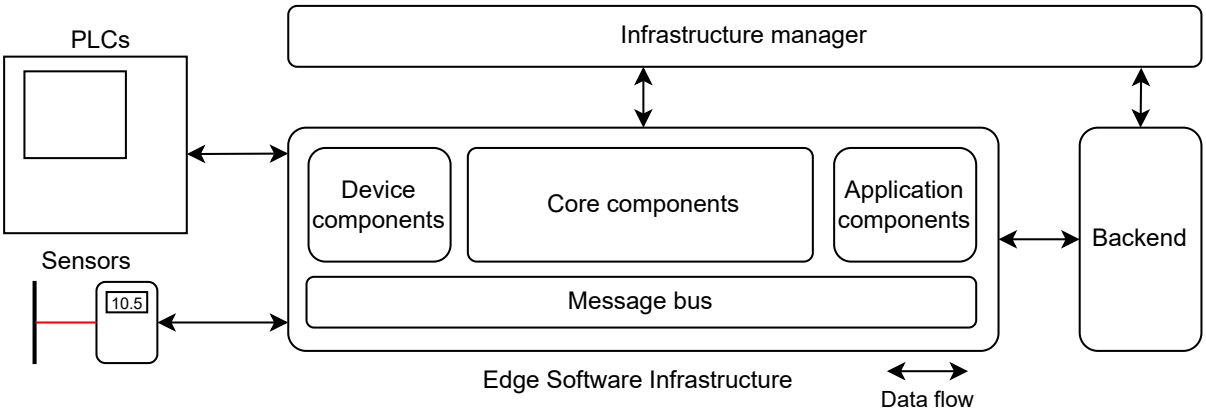


Figure 3.3: The figure depicts the different software components of the gateway.

Device components: The device components are responsible for enabling interaction between the gateway and external devices, such as PLCs and sensors, through a series of services. Their primary task is to collect data from the devices using their respective communication protocols and translate the incoming data into a unified data structure. Examples of communication protocols that may be utilised include: Modbus TCP, OPC UA, USB, and Snap 7.

Core components: The core components are what ensure the internal working of the edge platform. This can be done through components which manage connections, store data, maintain the configuration of the system and handle possible commands which has to be issued to the connected devices.

Application components: The application components can be used to perform a set of different tasks. Either they can be used to transform and process the data after it has been collected, for edge computing capabilities, they can also send the collected data to external systems or receive data from external sources while the system is running, such as variables or commands.

Backend software structure and components

The software components which will be a part of the backend software structure will be: Pub/Sub data distributor, data ingestion, data storage, data visualiser, and the possibility to add extra applications. The components for the backend and how they are structured are illustrated in Figure 3.4.

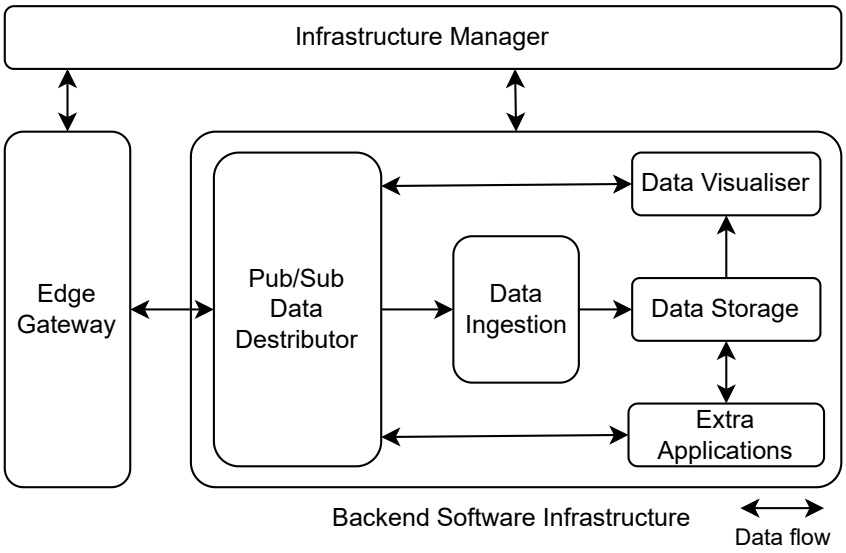


Figure 3.4: The figure illustrates the software components which will be utilised within the backend of the solution. Further, how the data flows between the components is also illustrated with the arrows.

Pub/Sub Data distributor: A crucial part of the solution is a component that handles data distribution, ensuring that data received from the edge is routed to the correct destinations. For this solution, it has been decided to implement a data distribution component based on the publish/subscribe (Pub/Sub) model. This approach allows new data consumers to be easily integrated into the system by subscribing to relevant topics. Various Pub/Sub data distributors can be used for this purpose, such as Kafka, MQTT, and Redis Pub/Sub, depending on the required features.

Data Visualiser: The data visualiser component of the backend will be utilised to host software used for data visualisation, making it possible for the user to access an interface where insightful graphs can be made, making it possible to monitor processes as data is produced and distributed by the broker. This component can consist of software such as Grafana or Kibana, which is made for visualising data, or it could also consist of custom-written applications based on libraries such as Plotly or matplotlib.

Data storage: The data storage component consists of one or multiple databases, where all the data extracted from the edge is stored, while also storing the data about the system. When looking into which database to utilise for the solution, there are many to choose from, such as MySQL, MongoDB, PostgreSQL and many more. As the data which is produced on the edge is time-series data, one kind of database which might be good to consider is time-series databases, which are databases made to handle this type of data. Some of these are TimescaleDB or InfluxDB.

Data ingestion: The data ingestion component will ensure that all the relevant data from the MQTT broker, which needs to be stored, is sent to the right location in the data storage. This data ingestion is used as a link between the MQTT broker and the data storage, as the broker often does not have the capabilities of pushing the data to the storage and data storage solutions often can not subscribe to MQTT brokers.

Extra applications: To extend functionality beyond the capabilities of the implemented components, it would be convenient to support additional applications. These applications or components would need to be developed on software platforms capable of communicating with the internal interfaces. This should not pose a problem, as many software platforms often have community-developed libraries for such tasks, provided that the interfaces are standardised and well-documented.

3.2.3 Infrastructure manager

The two prior sections have covered which software and hardware components can be utilised for the solution to solve the problem statement. To manage these components, a part of the solution will be the infrastructure manager, which aims to introduce ease of use and intuitive design into the solution. The infrastructure manager will be used as an interface to set up and manage the backend and gateways while ensuring the connection between them and the already existing technologies within processes. Some components which an infrastructure manager could consist of are a Gateway Configurator and a Device Service Configurator.

Gateway Configurator

The goal of making the solution easy to use is to accommodate possible users with limited knowledge of how to set up and configure gateways. Therefore, the feature of the gateway configurator is to assist with the configuration of the gateways. This configuration could be initiated through the infrastructure manager, by providing necessary context and information about the gateway desired to configure, which could include name, location, group and IP, etc. Then the infrastructure manager could check if any devices with the given information are already present to avoid duplicates, and then save the information while connecting to the gateway, to check if it has the necessary software components installed, so that it can be utilised as a gateway, and then configure it. An ideal feature would be that the infrastructure manager could automatically detect devices with the gateway software on it, so that the user only needed to provide context to the configuration. An illustration of the possible steps a gateway configuration could go through via the configurator is shown in Figure 3.5.

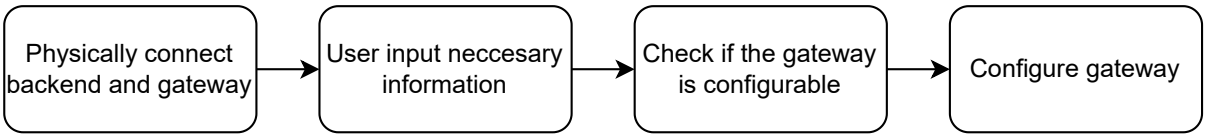


Figure 3.5: The figure illustrates possible steps that may be followed when configuring a new gateway.

Device Service Configurator

When the gateway has been configured, it needs to be connected and configured to extract data from the desired devices. To make the configuration and data extraction easy and intuitive, the device service configurator could have a Plug & Play feature implemented, assisting in the task of overcoming the knowledge gap of how to extract the data with the gateway. Ideally, this feature should make it possible to just connect the devices which produce data together with the gateway, and then it automatically knows how to extract the data and what kind of data is available. As this is an ideal scenario, which probably can be made with an extensive library of known devices, their communication protocols and also knowledge of what data is inside the devices, it will be too large a scope for this solution, a more realistic outcome would be a device configuration which is based on user inputs, but limited to information which is accessible for any user who might want to implement this solution. An illustration of the possible steps a device service configuration could go through via the configurator is shown in Figure 3.6.

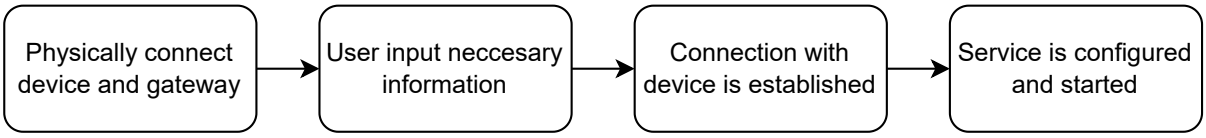


Figure 3.6: The figure illustrates possible steps that may be followed when configuring a new device service to connect to and extract data from devices within a process.

4. Implementation and Testing

This chapter presents the implementation of the proposed solution and evaluates its performance through three structured tests. Each test corresponds to one of the three phases the case study is divided into, as defined in the chapter 2. The goal is to verify that the solution behaves as intended and fulfils the requirements defined for each phase. The three tests are: Test 1 – Configuration and Deployment, configuration of the gateway and device services. Test 2 – Utilisation, seamless data access and process visualisation. Test 3 – Extensibility, flexible Sensor integration.

4.1 Implementation of the solution

This section will describe which components both hardware and software have been chosen to be implemented into the solution, while also how the components of the infrastructure manager work, from a user perspective. Afterwards it will be presented how the solution has been implemented into the screwing cell.

4.1.1 Choice of hardware components

The hardware components used for the solution will be based on what is available at the moment of development, but still is readily available parts that companies may already have or can easily acquire, as there are plenty of vendors and sufficient stock, which helps determine how available the solution will be to companies.

Edge hardware components
When choosing the hardware for the edge, it will determine the flexibility of the solution, regarding interoperability with already present devices. To adhere to the goal of being cheap, it is therefore important that the hardware component for the edge is not too expensive while still having certain capabilities, such as interfaces and computing power. The hardware acting as the gateway will be the SBC, Raspberry Pi 5 (RP5) with 8 GB of RAM, depicted in Figure 4.1. The reason why the RP5 was chosen was because of the availability, but also because of its price of approximately 140 € [27] while also being in stock at suppliers. Further, the RP5 also have a range of physical interfaces, some of which are GPIO, USB and RJ45, all of which are commonly used in the industry. Another good reason to choose an RP5 is how well supported and documented it is.

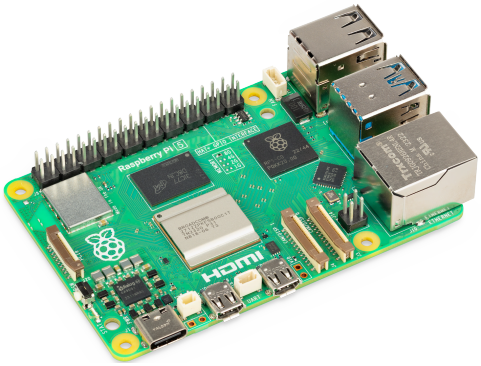


Figure 4.1: The Raspberry Pi 5, which is used for the hardware for the gateway. The image is borrowed from raspberrypi.dk [27].

The Raspberry Pi is no new SBC on the market and has been refined and upgraded through its iterations. It is also widely used by its community and even some places within the industry, which have implemented it into many different scenarios. Even though it might not have been through the same testing and standardisation as an IPC might have, the RP5 can still be said to be a well-tested piece of hardware.

Backend hardware components

The hardware used for the hosting backend of the solution will be a mini PC (Intel NUC10FNH) with 16 GB of RAM. This mini PC was chosen because, as mentioned, it was available at the time, but it can also mimic what SMEs might have available within their company without having to invest in new hardware to deploy the solution onto. The mini PC is in a compact form factor, making it ideal for not standing out and taking up a lot of space, while SMEs are in the stage of validating the benefits of digitally transforming their processes. The Intel NUC is depicted in Figure 4.2

Figure 4.2: The mini PC, Intel NUC, which is used for the hardware for the backend of the solution. The image is borrowed from www.avxperten.dk [28].

The system can always be upgraded to a larger on-premises server or, if desired, to the cloud, utilising cloud computing from some of the many vendors, removing the need for having to maintain the hardware and ensure the redundancy of the systems, but instead comes with other related costs. The mini PC used for this solution is no longer available on the market, but similar mini PC's with newer internal components can be bought for approximately 140 € on offer [29].

4.1.2 Choice of software components

As an important part of the solution is to ensure that it remains inexpensive to deploy, all software components will be based on open-source software. This approach reduces dependence on vendors and their costly products, as by using open-source components, there will be no licensing costs associated with utilising the software.

Edge software components

The software components chosen to be deployed on the gateway will be within the already mentioned set of components, which all utilise the messaging bus: device components, core components and applications components. Which implemented components each of these sets will contain is depicted and highlighted in 4.3.

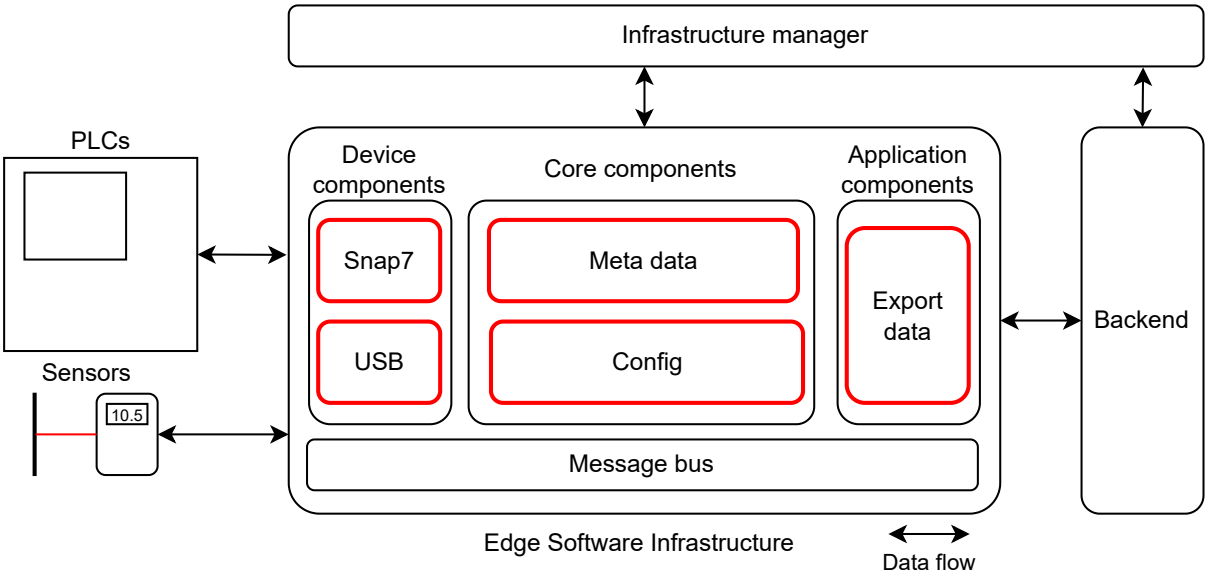


Figure 4.3: The figure depicts the different components, which will be deployed to the gateway. All components are connected with the internal messaging bus, ensuring seamless communication between them. The services make it possible to communicate with devices in a process and send collected data to the backend infrastructure.

Device components: The device components which will be a part of the solution an interact with the devices within the screwing cell will be Snap 7 and USB components. This is because it is possible to communicate with the PLC through Snap 7, which is the native communication protocol for Siemens PLCs, and USB will be used to add additional sensors.

Core components: The core components, which will be the backbone of the gateway, will be the metadata component, which contains all the data about the gateway, and the config component, which handles the configuration of the gateway.

Application components: The application components, for the solution, will be used to either send the states of the gateway or the collected data to external systems.

Backend software components

The components for the backend are: NanoMQ, which is an MQTT broker as the Pub/Sub data distributor, Grafana for the visualisation, and it will use the Python platform to develop any features which might be needed, such as data ingestion. TimescaleDB has been chosen as the data storage component, as it is a time-series optimised extension of PostgreSQL. This makes it well-suited for storing both high-volume time-series data from the processes and lower-volume contextual or configuration data. How the software components for the backend are structured is illustrated in Figure 4.4, where the implemented components are highlighted.

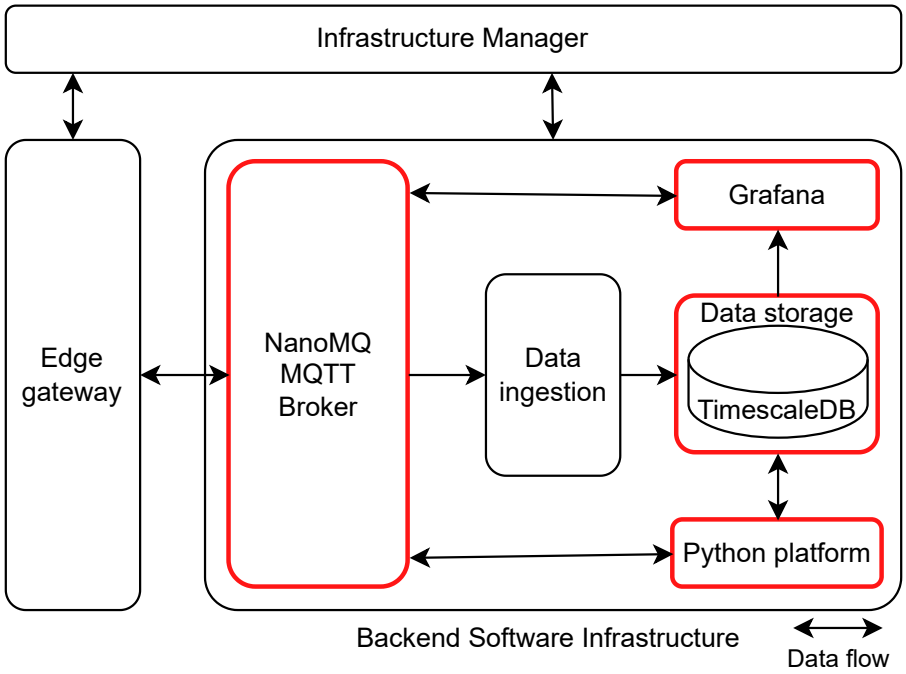


Figure 4.4: The figure illustrates the software components which will be utilised within the backend of the solution. Further, how the data flows between the components is also illustrated with the arrows.

NanoMQ: An MQTT broker has been chosen as the component for the data distributor for its simplicity, but also for its light weight compared to other Pub/Sub Data distributors. NanoMQ is specific because it is designed as a lightweight, high-performance MQTT broker for edge devices. Unlike alternative MQTT brokers, such as EMQX or HiveMQ, which offer open-source versions with restricted features alongside paid enterprise plans, NanoMQ is fully open-source without paywalled functionality. While it may not be as feature-rich as some competitors, its design reduces complexity and resource requirements, making it ideal for deployments on constrained hardware, which might be the only thing SMEs have available. Notably,

NanoMQ's performance is competitive with larger brokers [30], further justifying its use. As mentioned in Section 2.2.1, there are some standards which can be used in combination with the MQTT protocol to standardise elements of the protocol. In the case of this solution, some parts of the Sparkplug B standard will be implemented. What will be used is the standardised format of the topic namespaces and, to some degree, the messaging structure. The topic namespaces will have the following structure and elements:

$$namespace/group\_id/message\_type/edge\_node\_id/[device\_id]$$

The structure of each message with data will follow a JSON structure, with the components "timestamp" and "metrics", as illustrated in Figure 4.5. The timestamp will represent the time at which the message was published, and the metrics is a list which contains the data sampled from devices. The metrics component itself has components which describe the data, for this solution, 5 of the 10 available components from the Sparkplug B standard have been used: name, which is the name of the metric, timestamp, which is when the metric was captured, datatype, what datatype the metric has, value, the value of the metric which has been captured, and properties, can be custom data associated with the metric in this example the unit of the metric.


```
{
   "timestamp": 1744731241.7838410,
   "metrics":[
      {
         "name": "Sensor_1",
         "timestamp": 1744731241.7838407,
         "datatype": "float",
         "value": 16.3,
         "unit": "celsius"
      }
   ]
}
```


Figure 4.5: The figure illustrates how the messages with data sent from the gateway to the backend are structured.

The reason why the entire Sparkplug B standard is not implemented in the solution is that there are no mature open-source libraries that support easy implementation, particularly in areas such as message encoding and device behaviour on startup and disconnection. Another reason for not enforcing the Sparkplug B protocol across the entire solution is that it would prevent communication with systems that only support native MQTT, due to Sparkplug B's specific message encoding. However, using parts of the Sparkplug B standard, such as its approach to structuring messages and

defining topic namespaces, can still be beneficial for enforcing some level of standardisation. Additionally, the Sparkplug B specification states that certain messages, such as data messages from devices, should use a Quality of Service (QoS) level of 0. This means there is no guarantee of message delivery, which may not be desirable in all use cases.

Grafana: Grafana is an open-source software which enables querying data from different sources and visualising it in insightful graphs, making it possible to monitor processes live as data is produced and distributed by the broker. Grafana is used to solve the requirement of being able to visualise the data in the process, as it provides a feature-rich library of different ways to visualise data.

Data storage: TimescaleDB is a PostgreSQL extension designed specifically for time-series data, built on top of the PostgreSQL database system. This makes it suited for solutions that require both storage for high-volume time-series data and traditional relational data management. TimescaleDB enables efficient ingestion, storage, and querying of time-series data while retaining full SQL support. Its optimisations significantly improve query performance for time-based data and provide powerful features for real-time analytics.

Python Platform: To extend functionality beyond the capabilities of the implemented components, the Python platform will be used to develop extra applications. Python is known for being one of the easiest programming languages to learn and is therefore also a good choice when it comes to giving the option to develop components to people in SMEs who might be new to the field. In reality, any other programming language platform could also be used as long as it uses the interfaces of the other components, but they might have a steeper learning curve.

Data ingestion: The Python platform has been utilised within the solution to develop the data ingestion components, which ensure that the collected data gets stored within the database.

4.1.3 Infrastructure manager components

As mentioned, the infrastructure manager will consist of two components, the gateway configurator and the device service configurator, which aim to reduce the knowledge needed to configure gateways and connect them to devices to extract data. Each component of how they work and what is needed from the user to utilise them, to configure gateways and connect them to devices, is explained further.

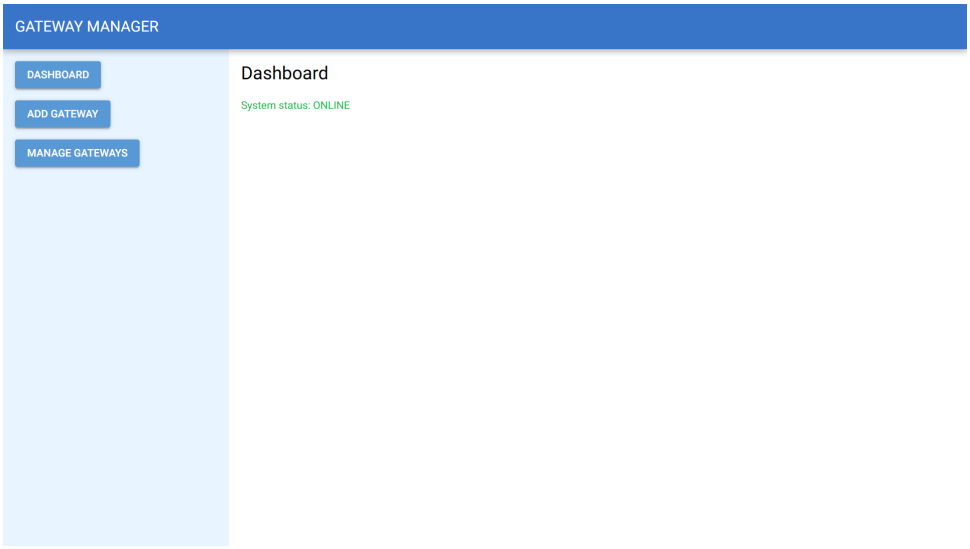


Figure 4.6: The figure shows the landing page of the infrastructure manager, which is also a dashboard, showing that the infrastructure manager is online.

When first entering the infrastructure manager, the user is presented with a dashboard informing the user that the system is online, as seen in Figure 4.6. Further, on the left side, there are 3 buttons which can be pushed: dashboard, which goes back to the landing page, add gateway, where new gateways are added and manage gateways, where the already added gateways can be managed. The figures of the infrastructure manager are presented in Appendix B, so to better get an understanding of how the infrastructure manager is designed, it is advised to have the appendix while going through the next two sections.

Gateway configurator

The Gateway Configurator component is used to connect to devices intended to function as gateways, devices that already have the required software components installed, and to configure them while establishing a connection between the gateway and the backend system. The gateway configurator is initiated under the "Add gateway" tab, which can be seen in Appendix B.2. It can be see that to configure a new gateway certain information is needed: The id which is desired to give the gateway, which group it is a part of, a description and the IP of the gateway, and then it need to be decided upon which connection the gateway will use to send data to the backend. The user can decide for themselves what the context inputs (ID, group and description) of the gateway should be, but has to know the IP of the gateway beforehand. When the information from the user has been inserted, and the add gateway button has been pushed, the gateway configuration sequence will begin, which is depicted in Figure 4.7. When the configuration is completed, the gateway will appear under the "Manage Gateways" tab, as seen in Appendix B.3.

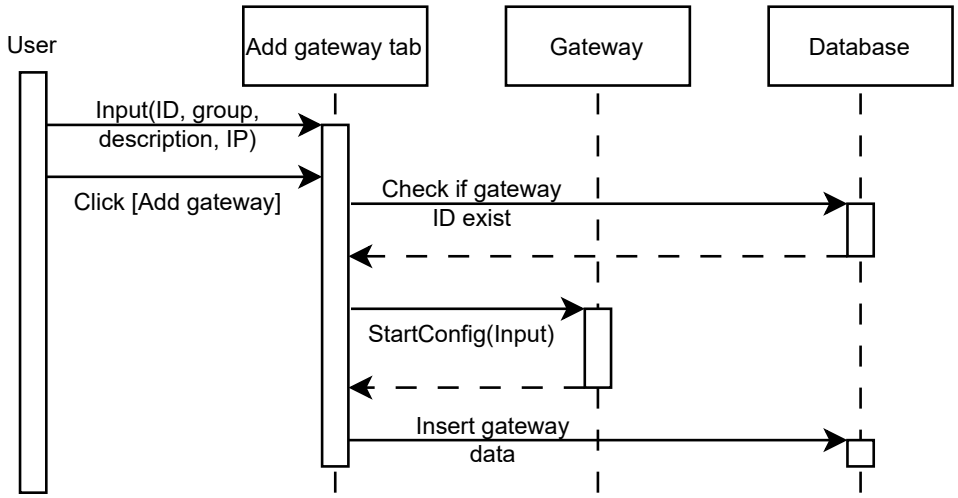


Figure 4.7: The figure shows the sequence of what happens within the infrastructure manager when a user is using the gateway configurator to configure a gateway.

Device service configurator

The device service configurator component is used to configure services on the gateway which connect to the devices to extract data. This could be the Snap 7 component, which can be configured to connect to Siemens PLCs. To access the device configurator, the user needs to click the "Add new devices service" button, which can be found by clicking on any gateway in the "Manage gateways" tab, as seen in Appendix B.4. When this button is clicked, a dialogue comes up where it is possible to determine: The ID, name, description, device type, and communication protocol to be used for the device service. When the combination of device type and communication protocol has been entered, the dialogue will present new required information which has to be entered by the user to connect to certain devices. When all the required information is entered, the user can push the "Add service" button and the device configurator sequence will begin, which is depicted in Figure 4.8.

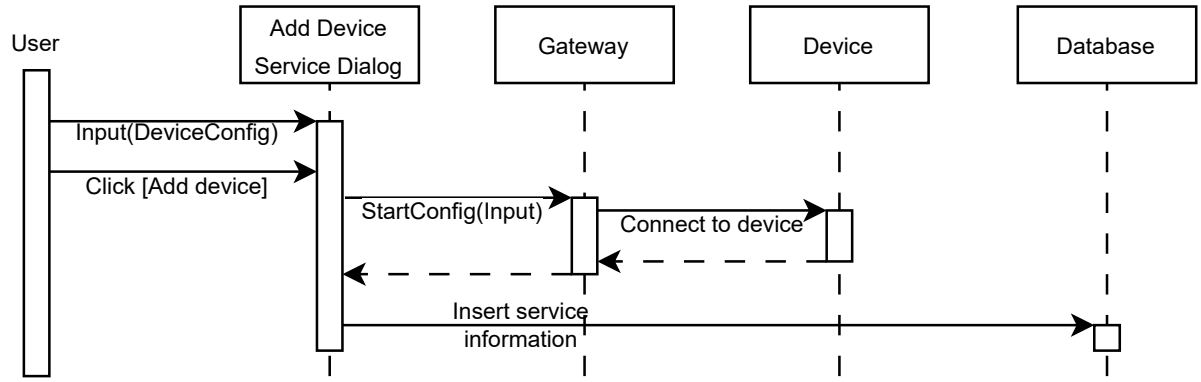
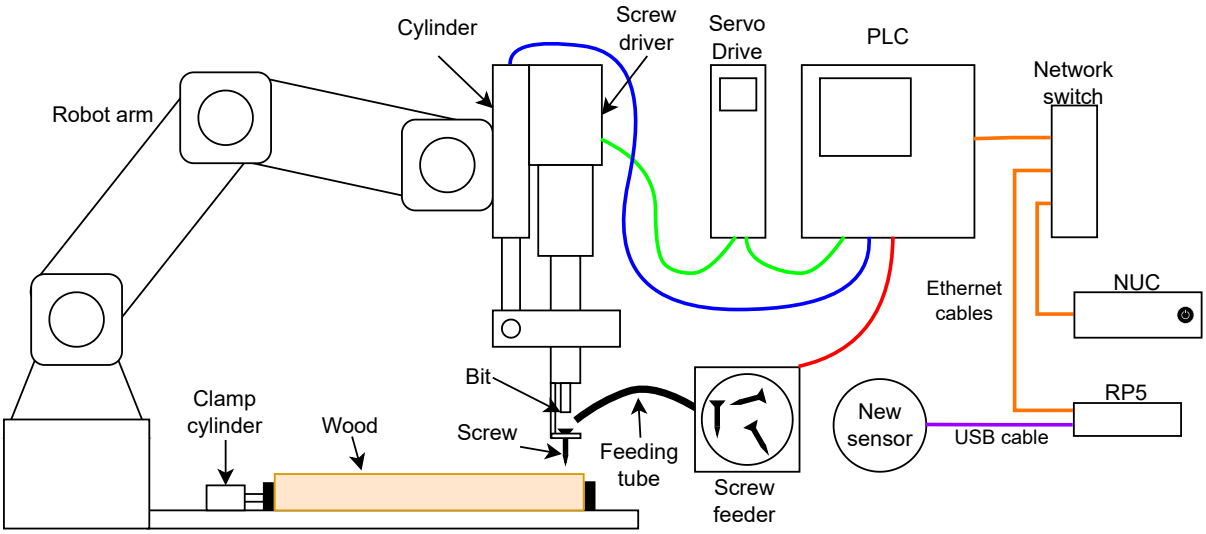


Figure 4.8: The figure shows the sequence of what happens within the infrastructure manager when a user i using the Device service configurator to configure the gateway to connect to a device.

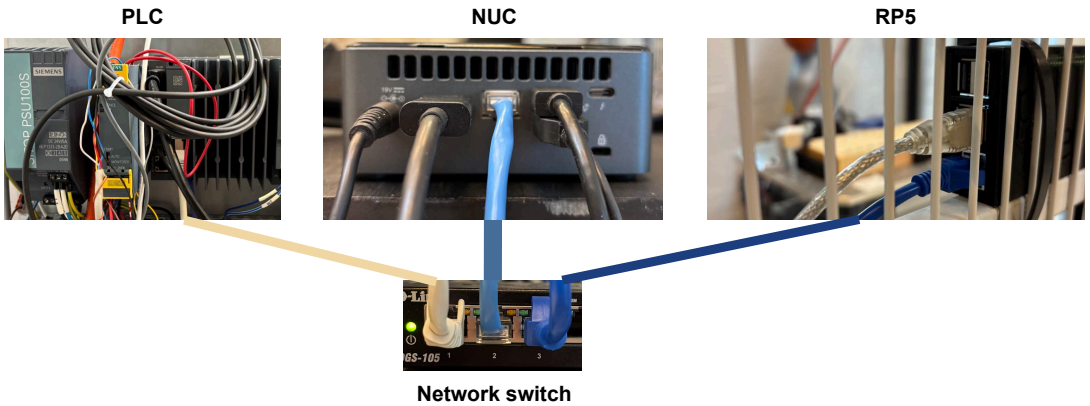
When the configuration is completed, the device service is visible in the page of the gateway as seen in Appendix B.4, and will begin to extract data based on how it is configured. For a full list of how different device services can be configured, and what data is necessary, look into the documentation available on GitHub [31].

4.1.4 Implementation in the screw cell

The hardware components of the solution and the screwing cell will be connected with each other through Ethernet cables, and since each device has a limited number of RJ-45 ports, a network switch will be utilised to create the connections. For the third test, where the addition of a new sensor is tested, the additional sensor, which will be added to the process, will be connected to the RP5 through a USB cable. A depiction of how the different components are connected is depicted in Figure 4.9.



(a) The figure illustrates how the setup is when the solution has been implemented



(b) This figure shows how the different devices are connected

Figure 4.9: Figure (a) is an illustration of how the solution is implemented into the process, while Figure (b) depicts how the different devices are connected.

The reason the NUC and the gateway are not connected wirelessly via Wi-Fi, despite the RP5 having that capability, is due to reliability concerns. In many production environments, a wired connection is preferred over wireless because shop floors can have significant electromagnetic interference that can disrupt wireless signals, leading to connection issues and potential data loss. Additionally, the way the network is configured at institutions like AAU makes wireless communications between devices more complicated.
The backend and infrastructure manager running on the NUC is not implemented inside of the screwing cell, but will be setup right outside the cell as it is used to visualise the extracted data and manage the gateway. This part of the setup will consist of the NUC with the backend software components pre-installed on it, and with a screen, keyboard and mouse connected to it, as depicted in Figure 4.10.

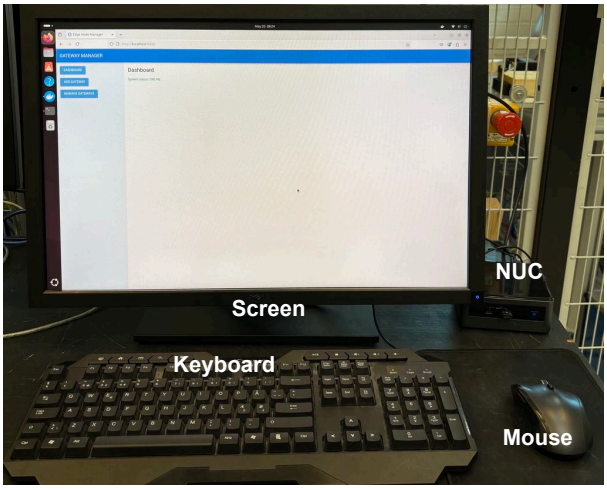


Figure 4.10: The figure shows the setup where it is possible to access the infrastructure and visualisation components.

This part of the setup will work as the interface between the user and the backend, where it will be possible to configure and manage gateways while also utilising the extracted data through Grafana.
To access the infrastructure manager and Grafana used for the visualisation, a web browser is needed. As the components are hosted locally, the URL address to put into the address bar will be "localhost:port" where the port of the address has to be changed depending on whether it is the infrastructure manager (The port is 8000) or Grafana (The port is 3000) which is being accessed. The two different web browsers are depicted in Appendix C.1 and B.1.

4.2 Testing the solution

The tests will be performed to determine if the solution can fulfil the requirements, having the essential functions of the solution and adhering to the final problem statement of being easy to use, having an intuitive design and being deployable on off-the-shelf consumer hardware. The order in which the tests will be performed is the same as the three phases which the case will go through, which also can be said to mimic how a company would implement the solution for the first time into a process, therefore, the test will be the following: 1. Configuration of the gateway and device services, 2. Seamless data access and Process visualisation, 3. Flexible Sensor integration. All the tests will be performed with the assumption that all the software components are already installed on the hardware, i.e. the tests will not tackle the tasks of how to install the software components onto the hardware. A guide to install the software components on both the gateway and backend is available on their respective GitHub page [32, 31]. Each test will aim to verify a number of requirements. How the solution performs in the test and how well it fulfils the requirements will be used as evaluation criteria for the solution. The tests will not validate all requirements, as some are validated through the design or chosen hardware. How the solution performs regarding the requirements will be further discussed in Section 4.4.

4.2.1 Test 1: Configuration of the gateway and device services

The first test focuses on the configuration of the solution on the hardware. Its purpose is to demonstrate that the infrastructure manager can set up and configure both the gateway and the backend using the implemented gateway configuration component, and connect a device to the gateway via the device configuration component, to extract the relevant data from the device. This process should require nothing more from the user than providing information about the devices involved, eliminating the need for the user to understand how to manually configure the connections, and therefore only requiring an understanding of the devices themselves and their content within. The test aims to verify the configuration and deployment requirements.

4.2.2 Test 2: Seamless data access and Process visualisation

The second test focuses on accessing the extracted data and visualising it. The purpose of this test is to demonstrate that the solution can successfully retrieve relevant data extracted from the process and present it in a visual format to provide meaningful insights. Using the configured device service from test 1, this test will verify

that data is extracted as expected by demonstrating how data visualisation can be performed using Grafana. This test aims to verify the utilisation requirements.

4.2.3 Test 3: Flexible Sensor integration

The final test involves integrating an additional sensor into the process and connecting it to the solution. The purpose of this test is to demonstrate the flexibility of the solution in the way that it supports communication with devices using different protocols and that it can be integrated into a process through both invasive and non-invasive methods. This test will make use of the previously validated device configuration component. However, the goal is not to revalidate this component, but rather to verify the system's ability to extract data from multiple sources in parallel. During the test, the gateway will simultaneously extract data from the PLC and the newly added sensor and forward both data to the backend. The new sensor used in this test will be a microphone, connected to the gateway via a USB interface. This test aims to verify the extensibility requirements.

4.2.4 Relevant available data

Before the tests can be conducted, it is necessary to identify which data is relevant to extract. In a Siemens PLC, data is stored in data blocks, where each data point is located at a specific byte offset. These offsets can be accessed via the Snap 7 communication protocol. The tests will be conducted under the assumption that the necessary knowledge about the data block structure and byte offsets is already available, allowing for straightforward data extraction. As previously described, the setup consists of a PLC connected to sensors and a servo drive. The PLC, along with the connected devices, already generates a range of process data. In this test, selected data will be extracted and visualised. The available data sources include:

Torque: The torque delivered by the screwdriver.

Angular Velocity: The angular velocity which the screw driver is rotating.

Angular Acceleration: The angular acceleration of the screw driver.

Angle: The angle of the screwdriver

Depth: The depth the screw has been screwed into the wood.

It may not always be necessary or relevant to sample data continuously once the gateway has been configured. There may be periods when the process is idle or

when the robot is moving between positions, during which no data related to the screwing process is being generated. In such cases, discrete signals from the PLC can be used to determine the system state and serve as triggers for data sampling or segmentation. These discrete signals include:

Process state: Has the process started

Screwing action: Has the screwing action started.

These state signals will be used to ensure that data is only sampled and visualised during the relevant parts of the process, specifically, when the process is running and the screwing action is taking place. This approach avoids capturing and displaying irrelevant data during idle periods.

4.3 Results

The results of each test will be documented, along with an explanation of how the tests were executed to achieve the given outcomes. Before the tests can begin, the software components for both the gateway and the backend must be installed on the respective devices. This will be done using Docker Compose, which enables the installation and deployment of software using a single configuration file and command. The installation process itself will not be discussed further in this report. However, if additional information about the software deployment is needed, it can be found in the project's GitHub repositories [31, 32].

4.3.1 Test 1: Configuration of the gateway and edge device service

As previously mentioned, the configuration of the gateway and the device service is carried out through the Infrastructure Manager. To complete this process, the step-by-step instructions available in the GitHub repository were followed.
First, a connection between the backend and the gateway was established, and the gateway was registered in the data storage component, enabling the backend to recognise that the gateway has been configured. This was done by entering the required context and IP address into the designated input fields, followed by clicking the "Add Gateway" button, as described in the guide. The interface for this step is shown in Appendix B.2. The information entered is tabulated in Table 4.1.

Field	Input
Gateway ID	AAU_ROBOT_GATEWAY
Group ID	group_1
Description	Robot screwing cell
IP	172.20.1.153
Connections	MQTT

Table 4.1: The table shows the information used to configure that gateway

As shown in Appendix B.3, the gateway was successfully configured and appeared in the "Manage Gateways" tab. The next step in the test was to configure a Device Service, which is the component responsible for connecting to a device within the process and extracting relevant data. In this case, the target device is the Siemens PLC used in the screwing cell. To add a device service for the configured gateway, the gateway was selected in the "Manage Gateways" tab, followed by clicking the "Add New Device Service" button. The guide on GitHub describing how to connect to a Siemens PLC was followed, and all the required information, such as the location of the desired data in the PLC, was entered into the appropriate fields. Once completed, the "Add Service" button was clicked to finalise the setup.
As seen in Figure 4.11, the configured device service appeared under the Device Services section. When its status was listed as "Online", it indicated that the service was correctly configured and ready to extract the specified data from the PLC. This confirms that both the gateway and the device service were successfully configured.

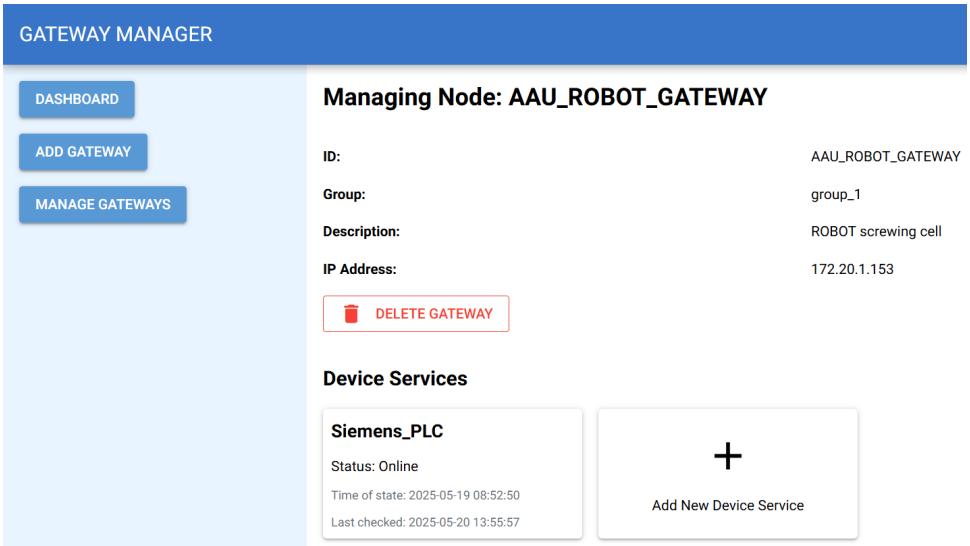
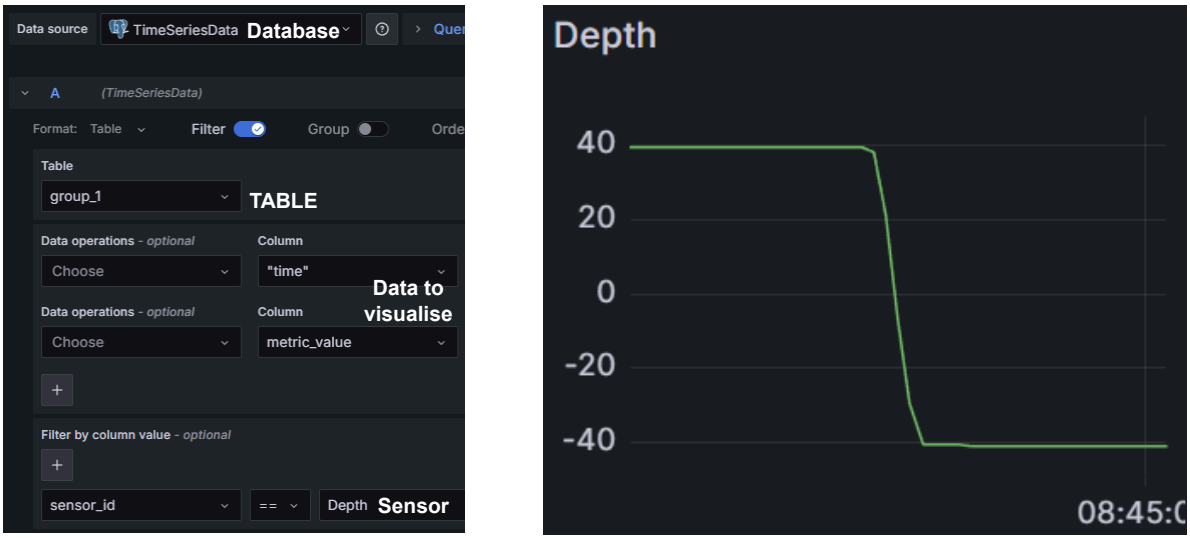


Figure 4.11: The figure shows the page where it is possible to manage the gateway, and that the configuration of the gateway and the device service for the PLC was successful, as they are present and online.

4.3.2 Test 2: Seamless data access and Process visualisation

Assuming the process is running, it can be verified whether data is being extracted correctly by checking its availability for visualisation in Grafana. To do this, Grafana's panel configurator is used to set up a panel capable of displaying the extracted data. The configurator allows for easy selection and processing of data using built-in features, and in cases where more advanced functionality is required, custom SQL (Structured Query Language) queries can be written. As shown in Figure 4.12a, a TimescaleDB table has been created for the gateway group. It is then possible to select and visualise data based on the identifier of each data source as seen in Figure 4.12b. This confirms that the data extraction process is working as expected. Panels can be created for each of the five data sources and are shown in Appendix C.3. Additionally, discrete state values extracted from the PLC, such as process state and screwing action, can also be visualised. These provide context for interpreting the data, as they indicate when the process is active and when screwing is taking place. An illustration of how the process states are visualised can be seen in Appendix C.2. It should be noted that the data shown in this illustration does not correspond to the same time period as the data presented in the five panels displaying the available data sources in the appendix. This demonstrates that the visualisation component was able to utilise the extracted data through visualisation, as intended.



(a) Configuration of visualisation panel

(b) Visualisation panel

Figure 4.12: The figure shows how the extracted data can be easily accessed and used to visualise the process.

One of the requirements this test is trying to verify is also the latency of the data transmission, the time from when a sensor sample is generated at the edge to when it

appears in the database. By comparing the sensor timestamp to the insertion timestamp, we observe an average transmission latency of 45.6 ms.

4.3.3 Test 3: Flexible Sensor integration

The additional sensor was integrated into the setup using a device component service, similar to how the PLC was configured. However, instead of selecting PLC as the device type, Sensor was chosen. Under the sensor type, Microphone was selected, and the communication protocol was set to USB. To configure the microphone, only two input parameters were required: the sampling rate and the number of channels to be recorded. To ensure that the microphone records only when relevant data is needed, its activation was tied to the state of the screwing process, mirroring the approach used for sampling data from the PLC. This was achieved by selecting from the list of available triggers, which are signals published by other devices on the internal messaging bus. In this case, the trigger was configured so that the microphone begins recording only when it receives the signal indicating that the screwing process is active. Once recorded, the audio data is transmitted to the backend, where it is stored and made available for further analysis or utilisation. The successful configuration of the additional sensor is shown in Figure 4.13, where the microphone appears in the device services section as "Microphone_1". Additionally, the presence of recorded audio files in the allocated folder within the backend confirms that data is being captured as intended.

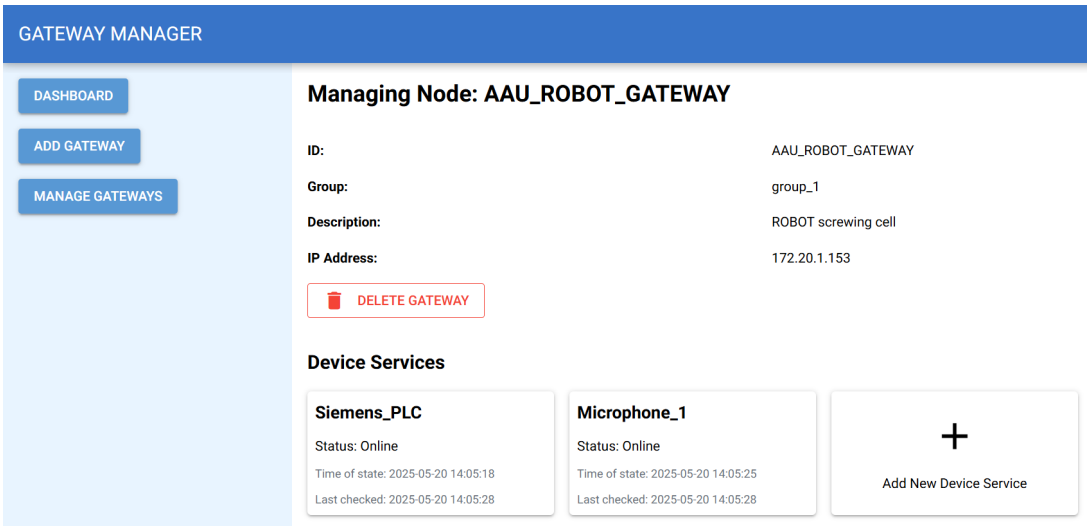


Figure 4.13: The figure shows that the solution can successfully integrate more sensors, such as a microphone which communicates over the USB protocol.

4.4 Evaluation

The evaluation will be used to evaluate if the solution has fulfilled the requirements through the implementation within the described process of the robot cell and the conducted tests. This section will go through how each requirement has been fulfilled by the solution and the choices taken throughout the implementation of it.

By choosing the hardware component RP5 for the gateway and the Intel NUC for the backend, it can be said that the following requirements are fulfilled:

- Requirement 6: By choosing to use the RP5 and a Mini PC, which are both off-the-shelf consumer hardware, the requirement can be said to be fulfilled.
- Requirement 7: If assuming that a company would have hardware already available within their premises to run the backend on. It is determined that choosing an RP5 as the gateway device fulfils the requirement of aiming to minimise the cost of deployment. Though there are many aspects to this which will be discussed in Section 4.5.
- Requirement 8: Since both the hardware and software, for both backend and edge, can be deployed on-premises, the requirement can be said to be fulfilled.
- Requirement 14: By utilising an RP5 for the solution, which provides over 3 physical interfaces, which was the minimum for the requirement, the requirement can be said to be fulfilled.

From how the software components of the solution are structured, the software components chosen, how they are deployed onto the hardware, and through tests 1 and 3, it can be said that the following requirements are fulfilled:

- Requirement 1: As MQTT was chosen to be utilised as the data distributor within the solution, it also provided the solution with one standardised interface used for internal communication, fulfilling the requirement.
- Requirement 2: As MQTT in combination with Sparkplug B, standardise how topics are structured while having open interfaces, making it possible to receive data just by subscribing to a topic, it fulfils the requirement.
- Requirement 3: Since the solution and its components are based on containerisation it makes it possible for the solution to be deployed on both SBCs and IPCs, as they both can utilise software such as Docker which is used to deploy containers, fulfilling the requirement of being able to be deployed on at least two of the 3 mentioned hardware (MCU, SBC or IPC).

- Requirement 4: By making the deployment of the solution possible through installing Docker and just deploying the solution, it removes the need for contacting external partners for getting the solution up and running, and thereby the requirement can be said to be fulfilled. Though the solution in its current state regarding the knowledge barrier will be discussed in Section 4.5.
- Requirement 5: As all the software components chosen for the solution are open sourced, it fulfils the requirements of only utilising open source software components.
- Requirement 12: In combination with the RP5 and the software components deployed on it, it allowed for adding an extra sensor to the setup, which was a microphone, which fulfils the requirement of the solution to be able to support adding additional sensors if needed.
- Requirement 13 & 15: By developing two device components, Snap 7 and USB, it fulfils the requirement of having at least two interfaces used for data collection while also fulfilling the requirement of supporting at least 2 industrial communication protocols.

From the execution and results of test 2, it can be said that the following requirements are fulfilled:

- Requirement 9: By looking at the timestamp of when the data from the device was sampled and when it was inserted into the database, it shows that the solution can ensure data availability and real-time insights, as it only on average took 45.6 ms for the data to be extracted and transmitted through the solution and thereby fulfilling the requirement of taking less than 100 ms.
- Requirement 10: Through the utilisation of the developed device components, it was possible the fulfil the requirement by extracting the relevant existing data within the process.
- Requirement 11: The implementation of Grafana as a visualisation component made it possible to utilise the extracted data through visualisation, fulfilling the requirement.

The implementation and testing of the solution show that it successfully meets the predefined requirements. Therefore, it demonstrates how a solution can be designed to support the digital transformation of industrial processes by extracting and utilising available data. Although this solution primarily focused on visualising the data, it lays the foundation for integrating more advanced Industry 4.0 tools in the future.

4.5 Discussion

The presented results indicate that the system performs as intended to meet the requirements. However, the current state of the solution still has limitations, which will be addressed in this section. Further, the deployment cost, the scalability and flexibility of the solution will be discussed, and lastly, the robustness and applicability of the solution will be touched upon to discuss if the solution could actually be implemented within a real process of an SME.

4.5.1 Limitations of the solution

Initially, the plan was to implement a "click-to-add" visualisation feature in the Infrastructure Manager to group all management features within a single application. However, due to resource constraints, this feature was not implemented. Instead, the existing capabilities of Grafana were used. While Grafana offers a powerful and flexible dashboard system, it can be overwhelming for new users who lack prior experience. Nevertheless, since Grafana is well-documented and widely supported, it was considered a suitable alternative to the originally planned feature.

Currently, the audio extraction component is limited to recording and saving audio files. As a result, visualising or directly utilising the audio data for analysis can be difficult. However, the audio data is available in the backend for further processing if needed in future implementations.
Another limitation surfaced when integrating the audio recording component. Initially, it was attempted to send the audio files using MQTT, but the communication protocol does not perform well with large payloads and is more geared toward sending many small messages. Although its specification allows for messages up to 256 MB, the system became congested when transmitting the audio files. To resolve this, HTTP was used instead to transfer audio data to the backend, which proved to be more suitable for this use case.

The Snap 7 device component also has some limitations. Currently, it can read data from PLC data blocks and use those values as triggers for sampling. However, the Snap 7 communication protocol also allows external devices to write values back to the PLC. This capability could be used to let the gateway start or stop processes on the PLC, which was not implemented in the current version.

Despite the low cost of the solution, there are still some practical barriers to deploying it. Setting up the solution requires knowledge of Docker and Docker Compose,

which may be challenging for users without prior experience. Additionally, when configuring the gateway through the Infrastructure Manager, users must supply the correct IP addresses. Depending on whether a wired or wireless network is used, IP acquisition can vary. In some scenarios, especially when connecting to a PLC via Ethernet, users may also need to configure network settings such as subnet masks or DNS addresses, tasks that can be daunting for non-experienced users.

4.5.2 Deployment costs

As previously discussed, the solution is designed to minimise deployment costs. The results demonstrate that it is feasible to deploy free, open-source software capable of meeting the system requirements on an RP5 and a mini PC. Although the RP5 is not the cheapest SBC on the market, it provides sufficient memory, computing power, and stability to support the solution. While cheaper SBCs may offer similar specifications, they often lack the same level of community support and reliability. Lower-cost, less resource-rich SBCs could potentially be used, but doing so might require redesigning or scaling down the solution to accommodate more limited hardware.
Additional cost savings are achieved by utilising existing hardware for the backend, such as a repurposed mini PC. If a company already possesses suitable hardware, deployment costs can be significantly reduced. Even if new backend hardware is required, it can be obtained at a relatively low cost. Overall, because the solution relies entirely on open-source software, the total deployment cost is primarily determined by hardware availability. In a typical setup using an RP5 and a mini PC, the total cost of deployment can be around 300 € [27] [29]. Therefore, it is said that even though the solution does not consist of the cheapest parts, it is still fulfilling the requirement of minimising deployment costs.
A key factor in deployment cost that must be considered is the human resources required to deploy the solution. Currently, this cost is not quantified, as it depends heavily on the knowledge and experience of the individual performing the deployment. This variability makes it difficult to provide a precise estimate. However, by reducing the number of manual tasks involved and streamlining the deployment process, as attempted with this solution, the required human resource effort can be significantly lowered compared to a scenario where a person must handle the entire setup and configuration by themselves.

4.5.3 Scalability and flexibility

The scalability and flexibility of the proposed solution will be discussed to highlight its possibility to scale, flexibility, supported features, and adherence to standards, compared to an ideal general solution briefly outlined at the beginning of the chapter.

As mentioned, the solution is tailored to the presented case study, narrowing its scope. Nevertheless, it retains significant scalability and flexibility through its software and hardware choices. For the gateway, the choice of basing the software architecture on a containerised component-based model, inspired by the open source project EdgeX Foundry, has made it possible to develop and add components as needed, making the gateway versatile in the sense that it is possible to scale and connect to a wide range of devices, as long as long as someone can develop the component handling the communication between the device and the gateway. The containerisation also allows the solution to be scaled to any hardware which supports the deployment of containers. The solution would be more general than it currently is if all communication protocols were already supported through developed components. This would allow it to connect to and support any desired device for data extraction. However, due to the scope of this report, support was limited to only two protocols. Additionally, by utilising an MQTT broker and a standardised database with open interfaces in the solution, it simplifies system integration and allows connectivity for new systems.

The choice of the RP5 as the hardware platform introduces certain limitations but strikes a balance between cost and general applicability. For example, the RP5 does not natively support industrial protocols such as EtherCAT, which would require additional hardware. It also lacks some of the physical interfaces typically found in industrial-grade equipment, such as analogue inputs/outputs or a serial port. Nevertheless, it provides the essential interfaces required for the case study and can be extended through "hats" or expansion boards, offering flexibility for other use cases.

Although the solution was tailored for the specific deployment presented in this thesis, its modular software and hardware design provide a degree of flexibility, making it adaptable to other industrial applications. That said, there are still open questions regarding scalability and potential undiscovered limitations when the system is deployed in different environments or with different hardware. Specifically, it remains to be determined how many devices a single gateway can support, how many gateways the backend infrastructure can handle reliably, and under which circumstances the backend needs to be scaled or an additional gateway would be required to maintain system performance.

4.5.4 Robustness and real world applicability

As previously mentioned, the environment in which the solution was tested does not necessarily reflect the conditions found in a typical SME production setting. Various environmental factors, such as humidity, dust, air quality, temperature fluctuations, oil or water splashes, vibrations, and space constraints, must often be considered when deploying solutions like this, as they can significantly impact reliability and performance.

If it is assumed that the backend system is deployed in a controlled environment, such as an office, where these conditions are not a concern, then only the RP5 unit would be exposed to the harsher conditions found on the shop floor. In the current setup, the RP5 is housed in a 3D-printed box. While this provides basic protection, it does not fully safeguard the device from potential hazards. However, thanks to its small form factor, the RP5 can be placed inside existing electrical cabinets, potentially the same cabinet that houses the PLC it connects to. In such cases, the device would be somewhat protected from oil or water splashes and vibrations, though it might still be affected by humidity, dust, extreme temperatures, and poor air quality.

The RP5 is capable of operating within a temperature range suitable for most industrial environments where humans are also present, so temperature is generally not a limiting factor. However, in scenarios where the RP5 must be deployed directly on a production line in a non-intrusive manner, it may be exposed to more aggressive environmental conditions. To ensure reliable operation in such settings, an industrial-grade enclosure with appropriate protection should be used.

For long-term deployments, however, it may be more appropriate to invest in certified industrial-grade hardware. While this may come at a higher cost, the improved reliability, potential warranty coverage, and reduced need for hardware replacement may justify the additional investment.

While most of these considerations relate to the hardware side of the solution, it is also important to assess the robustness and real-world applicability of the software components. The software components of the solution are generally suitable for real-world deployment, with the main challenges already outlined in the limitations section. A key requirement for successful implementation is having a basic understanding of where and how to access data within devices such as PLCs. If the SME has a technically skilled employee who is familiar with configuring such systems, it is feasible to deploy this solution within its limitations.

5. Conclusion and Future Development

This master's thesis set out to design and develop a solution that lowers the barriers small- and medium-sized enterprises face in their digital transformation efforts. The approach focused on ease of use and the use of affordable off-the-shelf consumer hardware components. To demonstrate the solution, it was deployed on a robotic screwing cell at Aalborg University, where its task was to extract and utilise existing process data while allowing for the integration of additional data sources.

The tests demonstrated that the solution is capable of extracting and utilising existing data from a PLC, enabling real-time insights into the process by visualising data produced by various components. Additionally, the solution supports the integration of extra sensors, such as a microphone, as used in the test case, allowing for the collection of further information. This illustrates that even in cases where a PLC provides only limited data, such as basic state information, the solution can still deliver valuable insights by mounting sensors in parallel with the process.
The use of a containerised architecture also proved effective, enabling new features to be easily added and configured as needed. This modularity makes the solution more adaptable and general for deployment in various settings. Furthermore, the implementation of a centralised MQTT broker allows easy integration of new components that require access to the process data.
While the solution demonstrated promising results, several limitations must be considered before it can be deployed in a real industrial setting. These include the environmental conditions in which the solution will operate, the need for a relatively high level of technical knowledge to install and configure the necessary software, the lack of long-term testing, and questions around scalability.
Overall, the results show that it is possible to lower barriers, particularly those related to resources, for small and medium-sized enterprises by leveraging open-source software and affordable consumer-grade hardware. A solution like this can help SMEs begin their digital transformation. By making it appear less daunting, easier to implement, and more cost-effective, the solution may encourage more SMEs to begin their digitalisation journey without relying entirely on external integrators or service providers. The solution's infrastructure manager and user interfaces allow for relatively straightforward configuration. However, further work is still required to enhance usability and reduce the technical knowledge needed for deployment. Given the current solution and its limitations, the potential for broader applicability, areas for improvement and suggested features for future development will be discussed in Section 5.1.

5.1 Future Development

As there still are many shortcomings which the solution has to include before being ready for a full fledged deployment within industries, and features which can be added so that it can become more general, making it possible to deploy in multiple scenarios, this section will discuss some of the areas, for future development of the solution to improve it and take it a step close to a finished product.

5.1.1 Communication protocols

As mentioned, a notable limitation of the messaging protocol MQTT used to distribute data within the solution is that it struggles sending larger files, such as audio files, even though it has a specified upper limit for the size of a message of 256 MB. Using MQTT for all the small messages, such as sensor data, will not be a problem, but suppose the solution was to evolve into utilising technologies, where a large amount of data needs to be sent to the backend for storage or analysis, such as images or video streams. In that case, another communication protocol or a change in how the messages for MQTT are structured is needed. If MQTT is still desired to be utilised, a feature can be implemented that, if the data which needs to be sent, e.g. a file, is above the specified limit, then it will break the data into chunks which fit within the specified limit and then distribute the messages.
As MQTT initially was not designed for video streaming or sending files, it could be a good idea to implement another communications protocol to send video streams if that is needed for future applications, as exemplified with the audio data from the test, which was sent over HTTP.

5.1.2 Computer vision and Machine learning

If a company wants to utilise and implement new areas within Industry 4.0 into their already existing processes, such as computer vision and machine learning, a Raspberry Pi might not be the best option, as there are other hardware options on the market geared more towards those areas. Therefore, if it is possible to test the solution and make it so that it can work and utilise the capabilities of hardware such as a Jetson Nano, the implementation of the solution could still have the relatively small form factor as the Raspberry Pi has, but now can tackle more intensive computer vision and machine learning tasks.

5.1.3 User friendliness

To further decrease the barriers which are present for SMEs when trying to digitally transform their processes, especially the knowledge barriers, there are still a lot of areas for improvement from the current solution. The current solution still requires the user to have a lot of knowledge about the devices that they want to be connected to, but also about the solution itself. For future development, the connection between the gateway and the devices it needs to be connected to can be improved by implementing a discovery feature, which can get all the information about the connected devices, so that the user does not need to have that knowledge. And for the connection between the infrastructure manager and the gateway, instead of having to know the IP of the gateway, a static, reserved IP can be implemented for new gateways when the software has been installed on the hardware. This makes it possible for the infrastructure manager to connect to an IP which is known to always be there for new devices, and then assign it a new IP when the gateway has been configured.
Further, it can be discussed how user-friendly the infrastructure manager is and its interfaces. Currently, it is split up into sections of the tasks that need to be performed, adding new gateways, managing the gateways and devices connected to the gateways. There might be areas of the design which can be optimised for user friendliness, such as where the different features of the infrastructure manager are positioned. Further, there is currently no type safety implemented into the infrastructure manager in the sense that when the user needs to input specific information, such as an IP, nothing checks if it is an IP or has the structure of an IP, which can lead to problems with connecting to devices.
For the current solution to be deployed on any systems, the user also has to have a certain knowledge about technologies which can be used to deploy containerised systems, such as Docker. This can be improved in the future so that the user only has to install one program and launch it once, and then everything is handled by the program behind the scenes, ensuring the setup is correct and everything launches in the correct order when the system is booted.
In general, user friendliness can be highly beneficial for users who understand the system and have a clear idea of what the expected outputs should be. For these users, features like automatic configuration and data extraction can eliminate much of the tedious setup work. However, for users who lack knowledge of the process or the expected outcomes, a system that is too user friendly, requiring little or no configuration, may offer limited value. If there is no one to assess whether the output is correct, the system's results cannot be validated. In such cases, a fully automated solution that removes the need for any domain knowledge can become ineffective, as there will always be a need for users who can interpret and verify its output.

5.1.4 Extra features

Currently, the number of features which the gateway, backend and infrastructure manager do not have as many features as other costly systems on the market, but it currently has enough features for performing the tasks within the scope of the thesis. To further increase the feature catalogue of the system, some possible features for development for each part of the solution are presented:

Gateway: Currently, the gateway can only be configured to communicate with the devices over the Snap 7 communication protocol and USB. Therefore, for the solution to be able to communicate with any of the many other PLC or sensor types, the number of communication protocols with which it can communicate with can be increased. Further features such as receiving commands to start and stop features, receiving data which can be utilised on the edge or a rule engine which ensures that certain tasks are performed when certain criteria are satisfied.

Backend: Currently, the backend only has one main feature for utilising the data, which is the visualisation of it over time. Though visualisation is a core feature of getting an insight into the process, there are still operations to the data which can be done to extract even more information from it. Therefore, to get other information from the data than just visualising it over time, features can be developed to calculate things such as overall equipment effectiveness, which can be used to identify how well equipment in a process is utilised, or utilise the data for predictive maintenance, so that the system can plan and schedule the next maintenance before any system failures might occur.

Infrastructure manager: As it is right now, the utilisation of the data is split into another application compared to the one which it used to manage and configure the gateways. This can be improved by creating one application which can handle both the utilisation of the data and management and configuration of the solution, so that a user can access everything in one place instead of having to know where things can be accessed, which can improve the overall overview and ease of use of the solution. Within a new application which can also visualise data, the kind of panels which can be used for visualisation can also be improved. For example, currently, for the audio which is recorded by the system, it is only possible to get the audio file. But if it was desired to also visualise this audio data, panels such as spectrograms could be used to get an insight into any anomalies that might be present in the audio of the process. Further implementation of a "click-to-add" feature, so that a user can easily add graphs and even include multiple kinds of data within the same graph, with a few clicks, so that correlations between what is happening in the process and the

outcome can more easily be drawn.
Another feature which can be useful for the infrastructure manager is to be able to get an insight into the state of each component within each gateway, and also each components of the backend, so that it is visible to see if processes might be idle or down unintentionally, or maybe some components need a restart to establish the connection between devices and the gateway.

Bibliography

Litterature

[1] *Short history of manufacturing: from Industry 1.0 to Industry 4.0*. URL: https://kfactory.eu/the-industrial-revolution-short-history-of-manufacturing/.

[2] Poonam Garg et al. "Development and Validation of an Instrument to Measure the Perceived Benefits of Digitalization in Manufacturing". eng. In: *IEEE transactions on engineering management* 71 (2024), pp. 8288–8306. ISSN: 0018-9391. DOI: 10.1109/TEM.2024.3390434.

[3] Sven-Vegard Buer et al. "The digitalization of manufacturing: investigating the impact of production environment and company size". eng. In: *Journal of manufacturing technology management* 32.3 (2021), pp. 621–645. ISSN: 1741-038X. DOI: 10.1108/JMTM-05-2019-0174.

[4] Dora Horvath and Roland Zs Szabo. "Driving forces and barriers of Industry 4.0: Do multinational and small and medium-sized companies have equal opportunities?" eng. In: *Technological forecasting & social change* 146 (2019), pp. 119–132. ISSN: 0040-1625. DOI: 10.1016/j.techfore.2019.05.021.

[5] Svend Aage Hansen et al. "Differences Between Small and Medium Sized Companies When Realizing Smart Production – Experiences from Northwest Smart Production Program in Denmark". eng. In: *The Future of Smart Production for SMEs*. Switzerland: Springer International Publishing AG, 2022, pp. 101–111. ISBN: 9783031154270. DOI: 10.1007/978-3-031-15428-7{_}9.

[6] Aoife Hanley et al. *The cumulative effect of due diligence EU legislation on SMEs*. Tech. rep.

[7] Mahdi Safa et al. "Enhancing Supply Chain through Implementation of Key IIoT Technologies". eng. In: *The Journal of computer information systems* 63.2 (2023), pp. 410–420. ISSN: 0887-4417. DOI: 10.1080/08874417.2022.2067792.

[8] Mohd Javaid et al. "Enabling flexible manufacturing system (FMS) through the applications of industry 4.0 technologies". eng. In: *Internet of Things and Cyber-Physical Systems* 2 (2022), pp. 49–62. ISSN: 2667-3452. DOI: 10.1016/j.iotcps.2022.05.005.

[9] Andreas Kornmaaler Hansen, Lasse Christiansen, and Astrid Heidemann Lassen. "Technology isn't enough for Industry 4.0 : on SMEs and hindrances to digital transformation". eng. In: (2024). ISSN: 00207543.

[10] Maria Stoettrup Schioenning Larsen et al. "Industry 4.0 Holds a Great Potential for Manufacturers, So Why haven't They Started?: A Multiple Case Study of Small and Medium Sized Danish Manufacturers". eng. In: *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems*. Cham: Springer International Publishing, 2022, pp. 721–729. ISBN: 303090699X. DOI: 10.1007/978-3-030-90700-6{_}82.

[11] Katharina Greger et al. "Technical Debt – Insights Into a Manufacturing SME Case Study". eng. In: *Perspectives in Business Informatics Research*. Vol. 529. Lecture Notes in Business Information Processing. Switzerland: Springer, 2024, pp. 191–206. ISBN: 9783031713323. DOI: 10.1007/978-3-031-71333-0{_}13.

[12] *How to Slash Technical Debt in Your Factory | Manufacturing.net*. URL: https://www.manufacturing.net/industry40/blog/22927254/how-to-slash-technical-debt-in-your-factory.

[13] *LEGACY Definition & Meaning - Merriam-Webster*. URL: https://www.merriam-webster.com/dictionary/legacy.

[14] Johannes Vrana. "The Core of the Fourth Revolutions: Industrial Internet of Things, Digital Twin, and Cyber-Physical Loops". eng. In: *Journal of nondestructive evaluation* 40.2 (2021). ISSN: 0195-9298. DOI: 10.1007/s10921-021-00777-7.

[15] *Modbus FAQ*. URL: https://www.modbus.org/faq.php.

[16] *Understanding Modbus TCP-IP: An In depth Exploration*. URL: https://www.wevolver.com/article/modbus-tcp-ip.

[17] *A Comparison of OPC UA and MQTT Sparkplug*. URL: https://www.hivemq.com/resources/iiot-protocols-opc-ua-mqtt-sparkplug-comparison/.

[18] *OPC UA vs. MQTT (a comparison of the most important features)*. URL: https://i-flow.io/en/ressources/opc-ua-vs-mqtt-a-comparison-of-the-most-important-features/.

[19] *Unified Architecture - Landingpage - OPC Foundation*. URL: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[20] *FAQ*. URL: https://mqtt.org/faq/.

[21] *Sparkplug™ Specification Version 2.2 Revision Number Date Author Description*. Tech. rep. 2019.

[22] *Microcontroller Boards*. URL: https://www.tinytronics.nl/en/development-boards/microcontroller-boards?sort=p.price&order=DESC.

[23] *Single-Board Computers*. URL: https://www.tinytronics.nl/en/development-boards/single-board-computers?sort=p.price&order=ASC.

[24] *Industrial Computers | Industrial PCs | RS*. URL: https://uk.rs-online.com/web/c/automation-control-gear/plcs-hmis-industrial-computing/industrial-computers/?sortBy=price&sortType=ASC.

[25] *Industrielle PC'er | Distributør af elektroniske komponenter DigiKey*. URL: https://www.digikey.dk/da/products/filter/industrielle-pc-er/1062?s=N4IgbCBcoJYCZRAWgIwAYUgDQgA4BdFsR8BPXAU0TgGcBjEAX0aA.

[26] *About - EdgeX Foundry Documentation*. URL: https://docs.edgexfoundry.org/4.0/about/.

[27] *Raspberry Pi 5 - 8 GB • RaspberryPi.dk*. URL: https://raspberrypi.dk/produkt/raspberry-pi-5-8-gb/.

[28] *Intel NUC BXNUC10I3FNHN2 Frost Canyon - Core i3 - 2xDDR4-SDRAM - 5459 DKK*. URL: https://www.avxperten.dk/mini-pc/intel-nuc-bxnuc10i3fnhn2-frost-canyon-core-i3-2xddr4-sdram.asp?srsltid=AfmBOopTw54ZuTBoDRuESYIhPjm_KBUlWqvBE_pfz2N87T9RpFky7ZVv.

[29] *NucBox G3 Plus–Enhanced Performance Mini PC With Intel N150 Processor*. URL: https://www.gmktec.com/products/nucbox-g3-plus-enhanced-performance-mini-pc-with-intel-n150-processor?spm=..collection_93170c86-021a-4673-944e-6b83c18ca778.collection_1.8&spm_prev=..index.header_1.1&variant=9be05933-1069-4464-81f7-246013e9dca7.

[30] *Open MQTT Benchmarking Comparison: EMQX vs NanoMQ | EMQ*. URL: https://www.emqx.com/en/blog/open-mqtt-benchmarking-comparison-emqx-vs-nanomq.

[31] *Jeppeotte/Infrastrcuture_manager*. URL: https://github.com/Jeppeotte/Infrastrcuture_manager.

[32] *Jeppeotte/PI_Edgegateway*. URL: https://github.com/Jeppeotte/PI_Edgegateway/tree/master.

A. Requirements description

1. The solution should provide at least 1 standardised interface used for internal communication, because a standardised interface ensures modularity and compatibility between different components or systems. This makes it easier to connect new systems easily, which can save resources.
2. The internal communication protocol must be standardised and open interfaced to ensure interoperability. Using a standardised and open communication protocol ensures that various components can interoperate regardless of vendor or origin, reducing vendor lock-in and enabling integration with other systems.
3. The solution should be able to be deployed on at least 2 of the 3 mentioned hardware types (e.g., MCU, SBC, or IPC). Flexibility in deployment hardware allows the solution to be used across a wider range of industrial scenarios, from simple monitoring to complex automation, also making it possible to fit both low- and high-end setups, and accommodate different price ranges.
4. To remove the need for relying on external partners, it must be deployable without external assistance. SMEs often lack access to integrators or IT departments. A solution that can be deployed independently increases adoption and lowers the costs associated with digital transformations.
5. To reduce the resource barrier, the solution must only use free and open source software components. Open-source tools lower the cost of deployment and prevent vendor lock-in, while also often can be customised as needed.
6. The solution must be able to be deployed on off-the-shelf consumer hardware, to ensure component availability. Readily available hardware, such as Raspberry Pi or Intel NUC, ensures easy access, reduces lead times, and keeps the cost low.
7. To further reduce the resource barrier, the solution must aim to minimise the cost of deployment. Budget constraints can be an issue for SMEs. A low-cost solution increases feasibility and potential for scale without needing significant capital investment. While it also makes it possible to prove the benefits of opening up for more possible investments.
8. The solution must be able to be deployed on-premises, to remove the need for external cloud providers. On-premises deployment can be preferred due to data privacy, latency, and internet connectivity, but it also makes it possible for the SMEs to not get locked in by any cloud providers, which might take a high price for storing and handling the data.

9. To ensure data availability and real-time insight, the extraction and transmission of data should not take longer than 100 ms. Timely data is crucial for real-time monitoring and quick decision-making. Therefore, keeping latency below 100 ms ensures data is actionable and while giving a real-time insight into a process.
10. To get the data from the already existing process, the solution has to act as a gateway which can extract data. Legacy systems and devices often have their proprietary way of exposing data, which might not be in a usable form. A gateway component bridges this gap, allowing data extraction from PLCs or other equipment without major modifications.
11. To be able to utilise the data, the solution has to distribute the data to a visualisation component. Distribution and visualisation make insights understandable and accessible to the SMEs, making it possible to respond faster and make process optimisations.
12. As there might be a need for adding extra sensors to the setup of the case, the solution must be able to support this. There might come times when new data i needed from a process, which requires adding an extra sensor. Therefore, allowing to add extra sensor ass needed makes processes adaptable and can evolve over time as the needs changes.
13. For the solution to be versatile in an industrial setting, it must support at least 2 industrial communication protocols. Different machines use different protocols. Supporting multiple ensures the solution can interface with diverse equipment and can be used in different scenarios.
14. For the solution to be more general, it should have at least 3 common industry physical interfaces (e.g. RJ-45, USB, GPIO). Physical interfaces determine what the solution can physically connect to. Supporting multiple standard interfaces increases compatibility and potential applications across industries.
15. The solution must have at least 2 standardised interfaces used for data collection. Multiple standardised data interfaces improve the flexibility and robustness of the collection of data.

B. Infrastructure manager illustrations

B.1 Dashboard

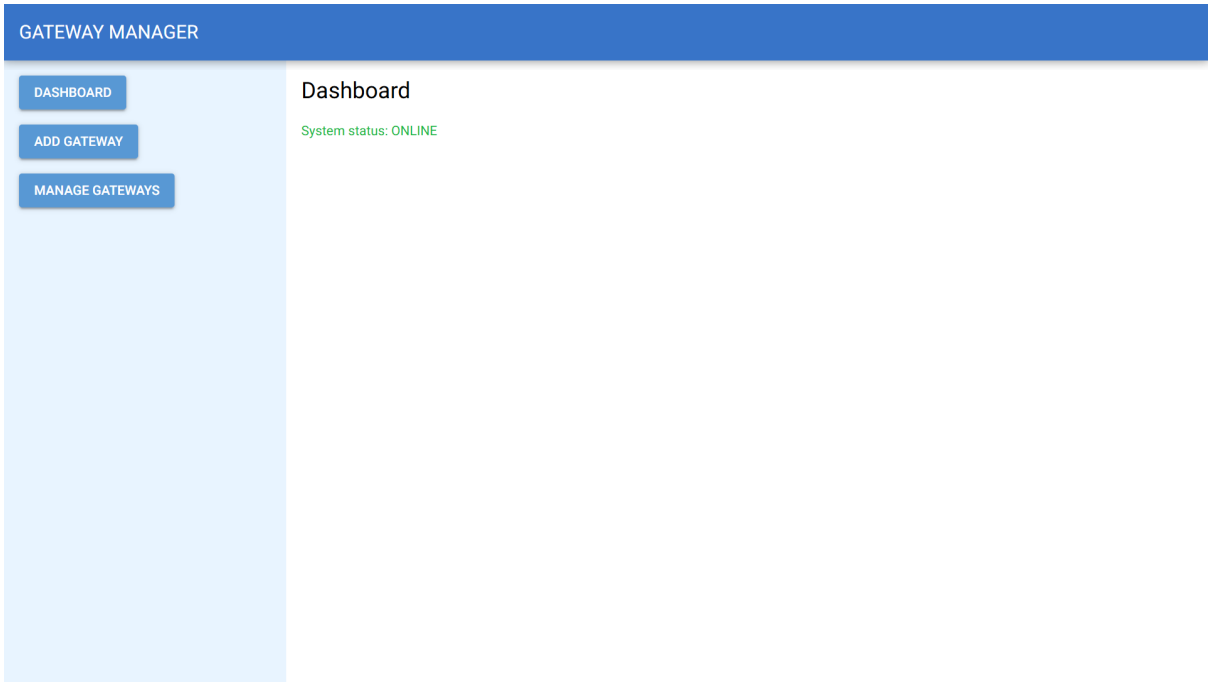


Figure B.1

B.2 Add gateway

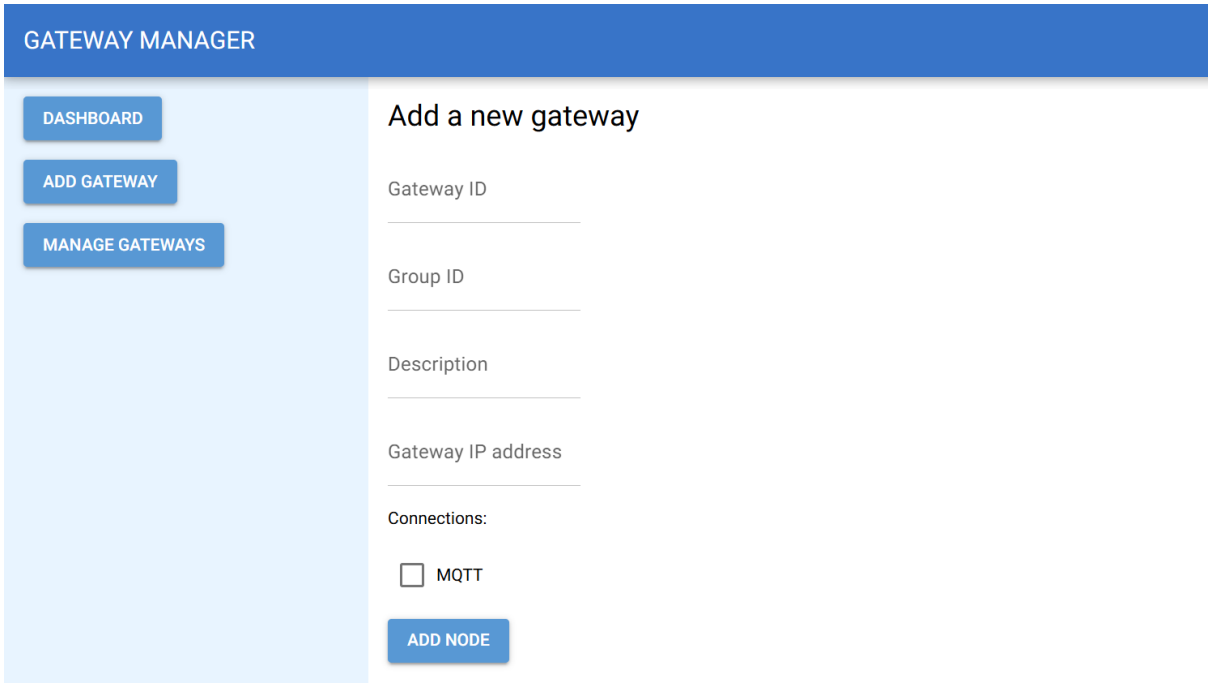


Figure B.2

B.3 Manager gateways

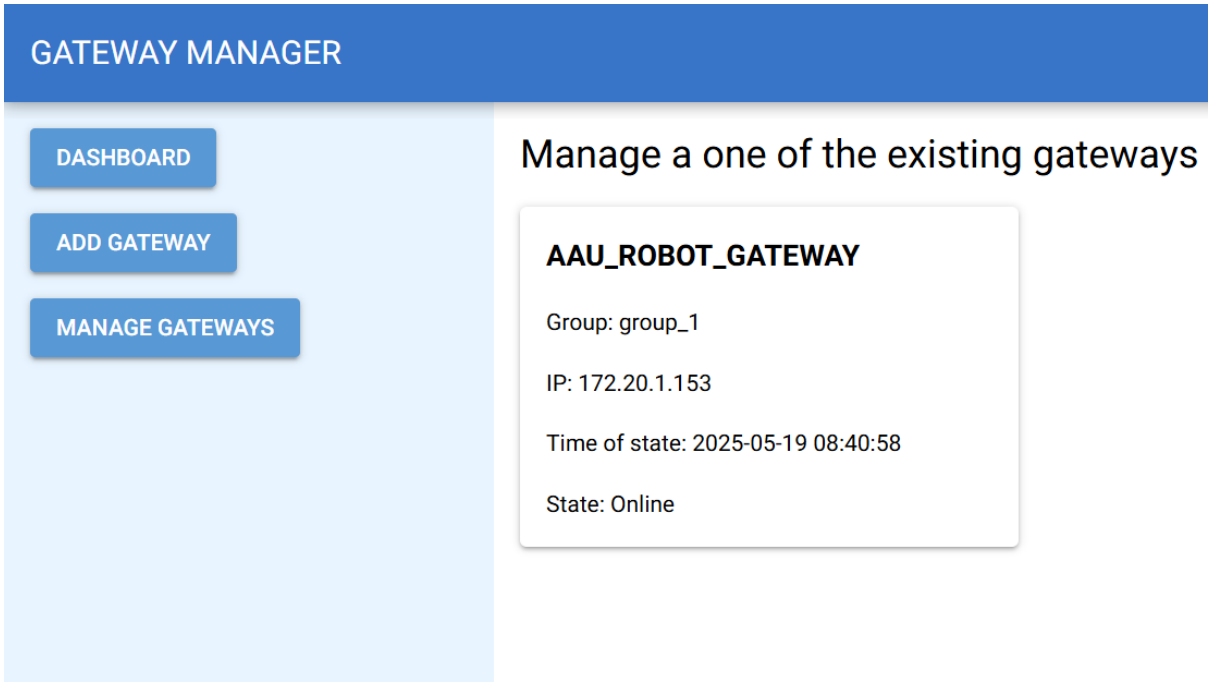


Figure B.3

B.4 Gateway manager

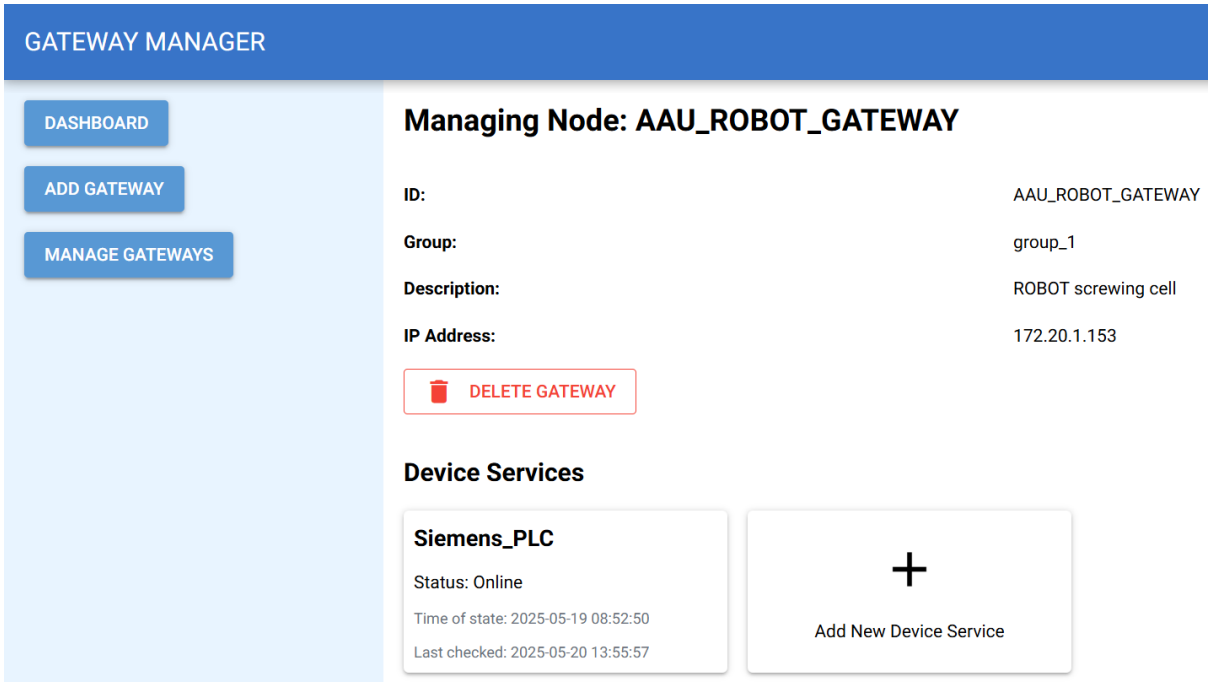


Figure B.4

B.5 Add new device

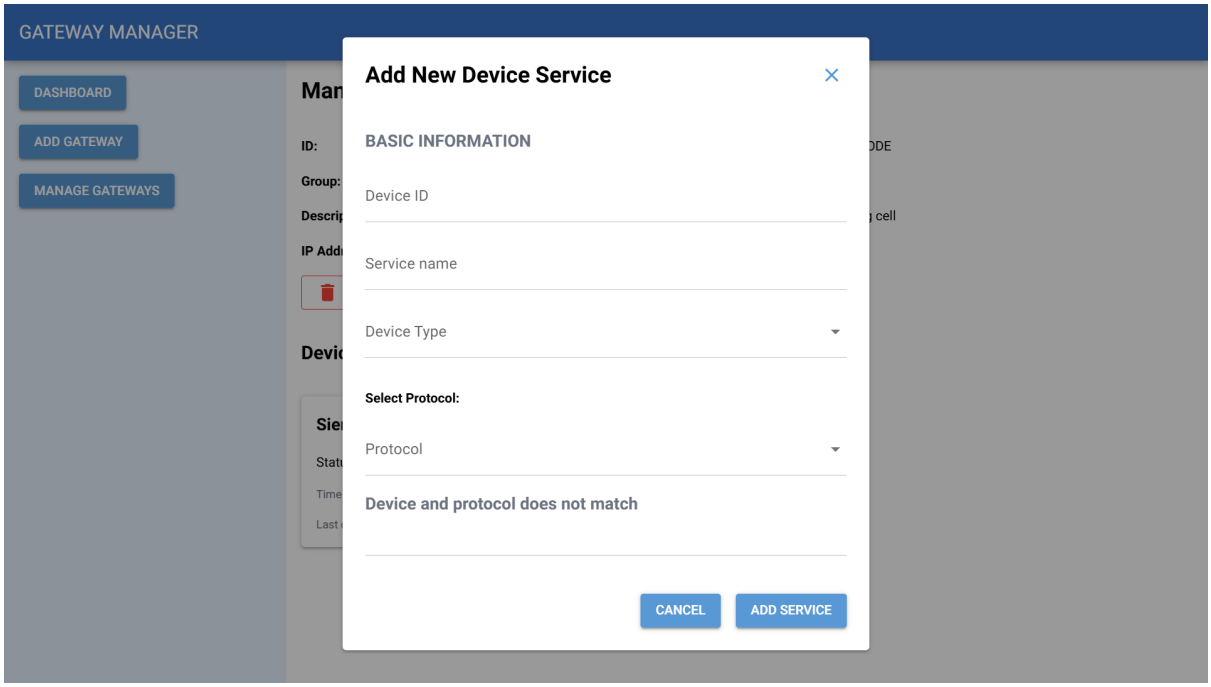


Figure B.5

C. Grafana Illustrations

C.1 Dashboard

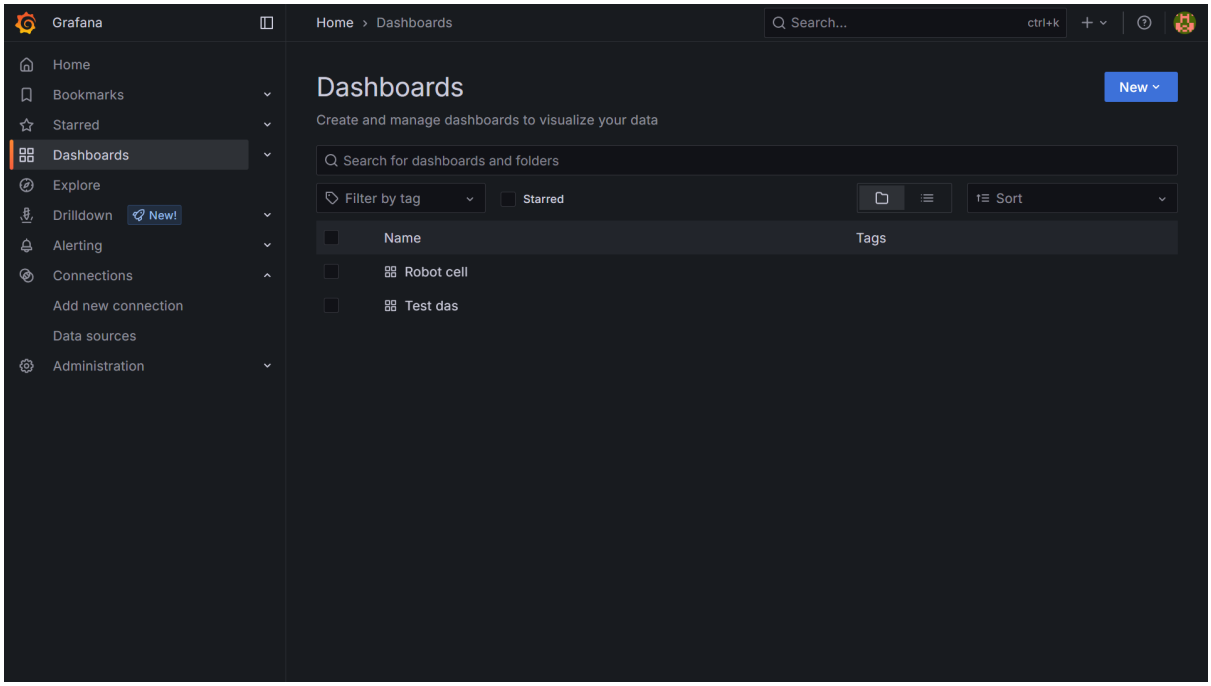


Figure C.1

C.2 Process states

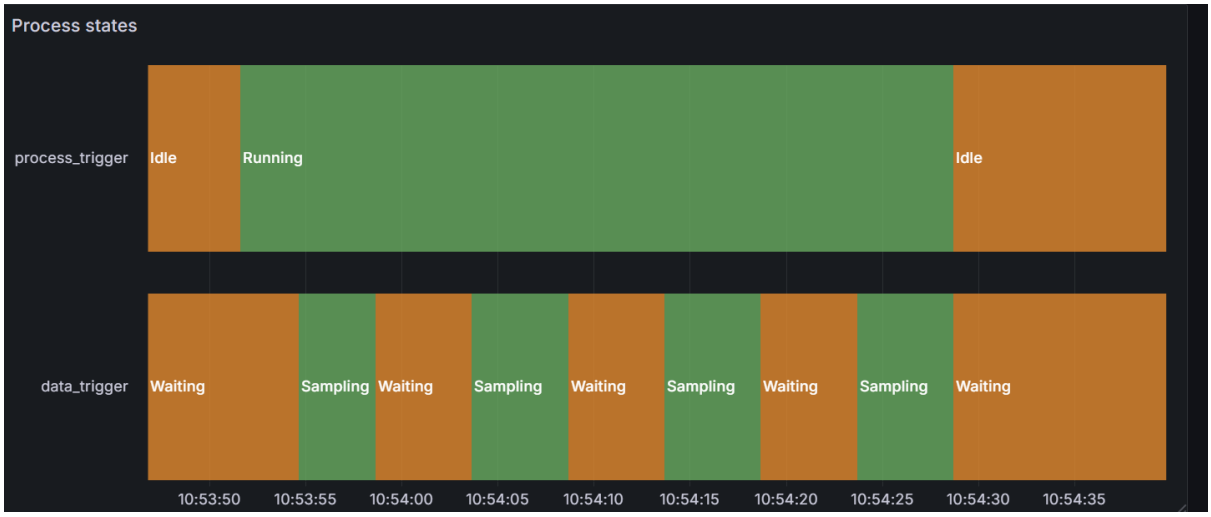


Figure C.2

C.3 Dashboard

Figure C.3