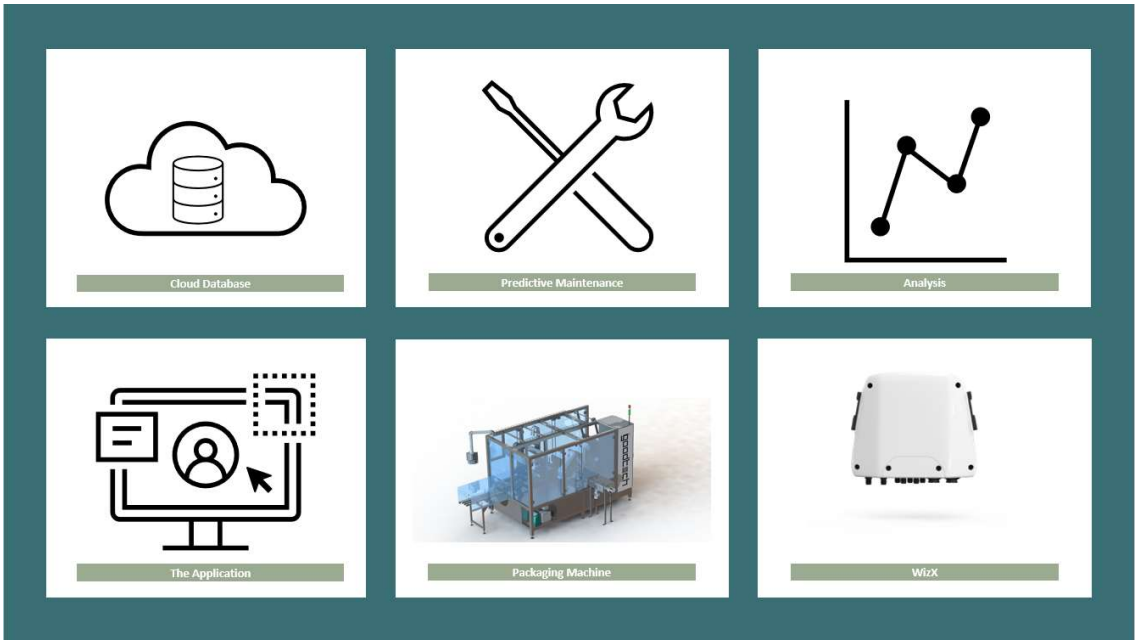
University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2022

Development of a predictive maintenance application for packaging machines



Peshawa Galali

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2022

Title: Development of a predictive maintenance application for packaging machines

Number of pages: 75

Keywords: Packaging Machines, Air Cylinder, Node-RED, Grafana, WizX, Machine Learning, Predictive Maintenance

Student: Peshawa Galali

Project Groupe: MT-78-22

Supervisor: Nils-Olav Skeie

External partner: **Goodtech As**

Summary:

The aim of this master's thesis was the continuation of the analysis and proof of concept of an application named (Application Packaging Machine) conducted in the fall of 2021 of a predictive maintenance application for packaging machines manufactured by Goodtech As in Moss, where the application was analyzed and proof of concept conducted. The application aims to monitor and detect early signs of wear and tear of the cylinders based on the life expectancy provided by the supplier and to perform regular maintenance on the servo motors based on the running hours recorded by the frequency drivers for the packaging machines.

Further application analysis is performed, including storage of data in the Microsoft Azure database for MySQL Server and security analysis on Azure. Following the analysis, the application was designed and implemented using Node-RED to collect and store data both locally and in the cloud. Grafana has been used to create dashboards that display data from the air cylinder, the VFDs, and the air pressure sensors. Additionally, alerts with email notifications are integrated into the dashboard for packaging machine data. The application was tested on the Test Celle machine located in the Moss manufacturing hall. Based on the application requirements, monitoring data from the Test Celle and storing them in the cloud was expected.

In addition, a suggestion for how Machine Learning could be applied in a predictive maintenance application for packaging machines is included.

Preface

It is the goal of this master's thesis to develop a predictive maintenance application for packaging machines manufactured by Goodtech As in Moss. The application monitors the air cylinder lifespan provided by the supplier, the level of compressed air pressure, and the frequency driver's operating hours to ensure timely maintenance of the servo motors in the machine.
Furthermore, a suggestion is made on how machine learning could be used in a predictive maintenance application for packaging machines.

Special thanks go to Jan Meen and Magnus Due for their time and efforts in transforming this case study into a school project. I would also like to thank Morten Halkjær for following up with me throughout the project and making sure that everything is in order.

Thanks to Stein Gunnar Holter Anders Asklien from Goodtech in the Moss department for their technical assistance regarding packaging machines and supervision. Thank you to Johannes Møgster and Trod van Delft for their technical assistance during this master's thises.

Thanks to my supervisor, Nils-Olav Skeie, for his excellent supervision and encouragement during this master's thesis.

Porsgrunn, 18th of May 2022

Peshawa Galali

Contents

Nomenclature

AL	Artificial Intelligence
API	Application Programming Interface
DBMS	Database Management System
ER	Entity-Relationship
GUI	Graphical User Interface
HMI	Human Machine Interface
I/O	Input/Output
IaaS	Infrastructure as a service
IIoT	Industrial Internet of Things
ML	Machine Learning
NIST	National Institute of Standards and Technology
OPC UA	Open Platform Communication Unified Architecture
PaaS	Platform as a Service
PLC	Programming Logic Controller
PM	Packaging Machine
PoC	Proof of Concept
SaaS	Software as a Service
SL	Security Level
SLA	Service Level Agreement
SSH	Secure Shell
SSL	Secure Sockets Layer
UFW	Uncomplicated Firewall
UI	User Interface
UP Model	Unified Process Model
VFD	Variable Frequency Drive
VM	Virtual machine
VPN	Virtual Private Network
WA	Wraparound

1 Introduction

The purpose of this chapter is to provide background information regarding the problem description for the master's thesis can be seen in Appendix A. An overview of the preliminary project completed in the fall of 2021 is included in this problem description. Additionally, the report's objectives and structure are presented.

1.1 Background

Industrial Internet of Things (IIoT), digitalization, and Industry 4.0 have attracted worldwide attention in recent years. Some organizations and manufacturing industries are already utilizing advanced networking, Machine Learning (ML), data analytics, robotics, 3D printing, and other technology to enhance their daily process and reduce dependence on human labor and decision making. Goodtech AS is focusing on the advantages of these technologies to meet the needs of their customers. In Moss's department, Goodtech AS builds machines and lines for handling packaging in the food and pharmaceutical industries [1]. These packaging machines include, among other components, different types of pneumatic air cylinders and Variable-Frequency Drivers (VFDs) that drive servo motors. The pneumatic air cylinders have an operational life span like any other components. After it has been in operating for a number of strokes or a curtain distance of strokes in km, it needs to be replaced. The servo motors also need regular maintenance, such as lubrication, depending on how long they have been in operation. The maintenance of the packaging machines is strategized, created, and followed by the customer's service departments within a packaging machines owner. Today, the pneumatic air cylinders are in operation until their operating life service is up. When one or several cylinders reach their life service, a part of the machine or entire packaging machine may shut down. And then, if spare air cylinders are not available, it might take longer to restore the machine.

As is described in the project task description in Appendix A, the proposed application monitors the air cylinder's life service based on the cylinder supplier's approximate service life value and will notify the operators in advance before it is time to replace the air cylinders. By monitoring the operation time (running hours) from the VFDs, the maintenance of the servo motors can be controlled and better planned. Thus the unplaned shutdowns of packaging machines are being reduced. Furthermore, by implementing this application, by continuously monitoring the packaging machines, the maintenance of the air cylinders goes from regular inception to continuous monitoring.

Goodtech has delivered many packaging machine solutions to the food industry in Scandinavia for many years. There is a high demand for packaging machines on the market today, which is why Goodtech is interested in investing more in this area. As a service product to Goodtech's packaging machines, this application will make them more attractive, more accessible to sell, and provide Goodtech with a competitive advantage over its competitors. Other companies like Tronrud Engineering and Pito Pallpack also deliver packaging machine solutions in Norway. Still, they do not have an application for monitoring the packaging machines for protective maintenance purposes.

The application's analysis and proof of concept (PoC) were carried out as a preliminary project in 2021. This master's thesis will be a continuation of this project, including some more analysis, design, implementation, and testing of the application.

1.2 Summary of Preliminary Project

In the fall of 2021, the FM4017 Project course was used as a preliminary project for this master's thesis [2]. In the preliminary project, an analysis of the application, including the collection of requirements, was carried out. Based on the analysis results, a PoC was developed to ensure that the application would be feasible. This was accomplished by creating an application that generated numerical values to simulate the air cylinder's distance of strokes and running hours from the VFDs operating in the packing machines. A simple dashboard was created using a visualization tool to display the simulated data in the form of a chart and numerically. As part of the PoC, alerts were configured and tested for the simulated data. The result of the PoC was as expected and successful.

A review and analysis of four literature reviews on cloud computing security were conducted. NIST and IEC 62443 have been evaluated on the subject of data security. Different security methods are discussed that can be implemented in the application to secure data and the application. A brief definition of ML and a discussion of how it can be applied to the application to predict the maintenance of packaging machines were also presented.

1.3 Objective

This master's thesis is a continuation of a preliminary project completed in the autumn of 2021, in which the application was analyzed, and a PoC was established. This master's thesis aims to expand upon the analysis and complete the application with further necessary analysis, design and implementation, testing, and deployment phases. The subtasks for the master's thesis are set out in Appendix A of the task description.

1.4 Report Structure

Chapter 1 – **Introduction** Background, the objective of the master's thesis, and the report structure is presented.

Chapter 2 – **System Description** This chapter presents the description of components for developing the application. Also, the brief description of packaging machines and air cylinders.

Chapter 3 – **Methods** This chapter includes an overview of the software development process and a description of the process model that Goodtech AS has followed. ML methods and algorithms for predictive maintenance within this application are described.

Chapter 4 – **Analysis and Design** This chapter presents a brief summary of the prior analysis done and further analysis during this master's thesis. The design phase of the application is also presented.

Chapter 5 – **Results and Testing** This chapter presents the results based on the implementation and testing of the application.

Chapter 6 – **Discussion**

Chapter 7 – **Conclusion**

2 System Description

This chapter presents the system description for the Application Packaging Machine (PM) by first describing the existing system with its building blocks which contain a data collector system with its software components, a local database, and a data visualization tool for presenting data. This chapter presents a brief description of the packaging machines and a description of pneumatic air cylinders and how they function. Finally, an introduction to the [1]Application Packaging Machine as an extension to the existing system is described and presented with a system sketch.

Figure 2.1 illustrates the system sketch of the existing system collecting data from the air cylinders and VFDs via PLC operating on the packaging machine and any other sensor that is connected directly to the Data Collector system. The collected data are then stored within the local database and visualized and monitored using a Data Visualization tool. This application aims to monitor the air cylinder's distance of strokes status based on the cylinder supplier's estimated service life distance of strokes and notify the operator in advance when the distance of strokes is close to the provided distance of strokes from the supplier. In addition, by monitoring the operating time (running hours) from the VFDs, maintenance of the servo motors can be coordinated and better planned.

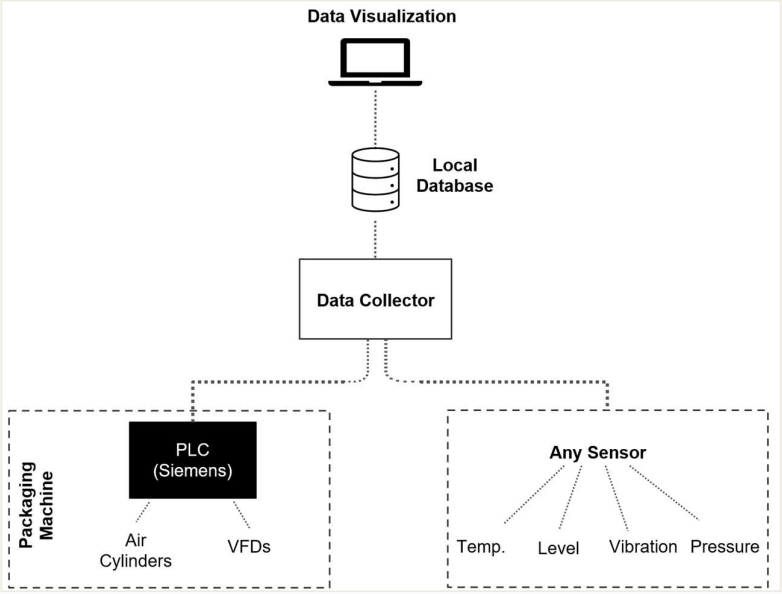


Figure 2.1: System sketch of the existing system [2].

[1] Application Packaging Machine is the name of the application developed in this master's thesis. This report uses the word sentence (the or this application), which refers to Application Packaging Machine.

2.1 Packaging Machine

The Moss Department of Goodtech As manufactures packaging machines for the pharmaceutical and food industries[1]. These packaging machines, among other components, include air cylinders and servo motors driven by VFDs which are controlled by a Siemens PLC of type Simatic S7-1500. The department builds a variety of types of packaging machines, such as the FlexPacker, FlexWrapper, and Wraparound. In this report, the functionality and area of use of a Wraparound (WA) packaging machine will be briefly described as the use and functionality of air cylinders on such machines.

The manufacturing hall in Moss has delivered many WA machines to companies in the food industry, including Orka Foods Norge and Insola. With Orka Foods Norge, packaging machines are used to pack and handle liver pate cans [3]. With Insula, packaging machines are used for the handling and packaging of fish cakes. Pharmaceutical manufacturers also use WA Machines for handling and packaging medicines. Figure 2.2 shows an example of a WA machine - a small machine with a limited amount of production space [4]. It is used to wrap around the packing in sizes up to 300 x 400 mm at a speed of 10 to 12 boxes per minute.

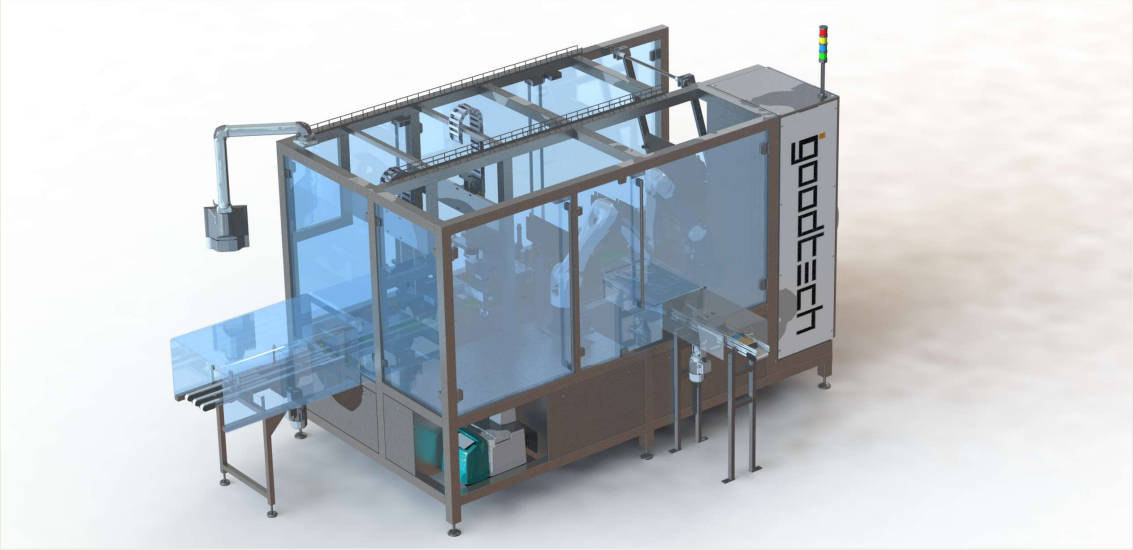


Figure 2.2: Wraparound (WA) Packaging Machine [2].

The WA machine is constructed using stainless steel, which is able to handle hard environments and is resistant to corrosion. Different types of pneumatic air cylinders and servo motors are used within the WA machine during its operation. A PLC controls the machine, and central compressed air is used to supply air to the cylinders. The WA machine contains different components that use air cylinders and servo motors during their specific performance on the machine. The main components include two KUKA robots, Tracker, the Turning Station, the Cardboard Magazine, and the Sheet Picker, as illustrated in Figure 2.3. The two robots are equipped with interchangeable tools that use two air cylinders to adapt to various package formats such as F-packs [3]. The tracker, also shown in Figure 2.3 (d), consists of a belt conveyor with an approx. take-over point one meter beyond the long edge of the packaging machine driven by a servo motor.

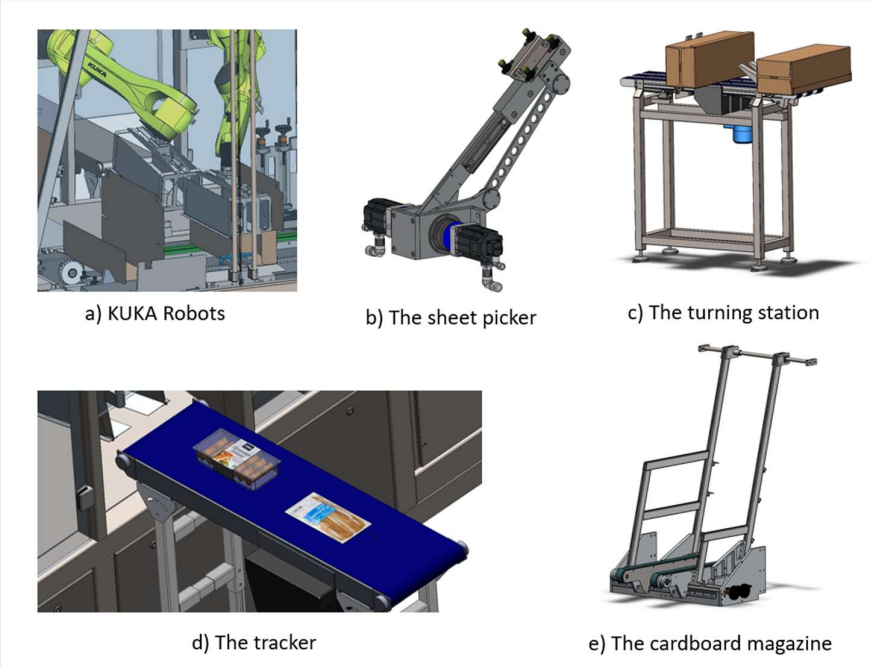


Figure 2.3: WA machine components. a) KUKA Robots. b) The sheet picker. c) The turning station. d) The cardboard magazine. e) The tracker [4].

As seen in Figure 2.3 (e), the cardboard magazine is another machine component that consists of two parallel conveyor belts driven by an air cylinder and guides the cardboard to the retrieve position for sheet pickers. The machine's sheet picker operation is controlled and driven by servo motors. At the outlet of the machine, there is the turning station component, as illustrated in Figure 2.3 (c), that turns the cardboard to 90 or 180 degrees using the air cylinders before stacking them on the pallet to be ready for delivery. The packaging machine can operate automatically and manually from the Human Machine Interface (HMI) mounted on the control cabinet door attached to the machine. An industrial signal tower stack security lamp light is mounted on the side of the machine, as shown in Figure 2.2. This lamp provides a warning, status, and alarm signals generated by the PLC if something is wrong with the machine during the operation.

2.2 Pneumatic Air Cylinder

Packaging machines involve several types of pneumatic air cylinders, each used for a specific task during packaging, such as cylinders with the piston rod (double-acting cylinders) and rodless cylinders [2]. Depending on the manufacturers, they are offered in various shapes and sizes where they are designed to be used for different purposes. Goodtech uses air cylinders from Festo, and an example of a cylinder with the piston rod type and piston rod double-acting cylinder can be seen in Figure 2.4. Like any mechanical component, air cylinders have operation life expediency that is based on either the number of strokes or the distance of strokes in km.

System Description

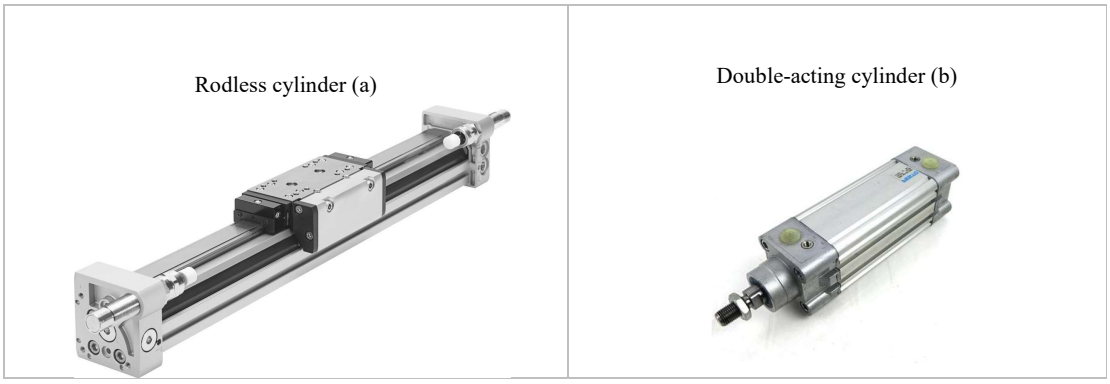


Figure 2.4: (a) Rodless cylinder from Festo. (b) double-acting cylinder from Festo [2].

However, service life can be shorter than specified by the supplier due to factors such as improper mounting style, overloading of cylinders with pressure, load, and energy, insufficient cleaning of cylinders from dust or splashed water or other fluids, and the environment in which the cylinder will be used, such as conditions of extreme temperature. If the factors outlined above are taken into account, it is possible to prolong the operation life expectancy of cylinders.

The pneumatic air cylinder is a linear actuator powered by compressed air to supply force on a load. Figure 2.5 illustrates the system sketch installation and signal process flow diagram from a double-acting cylinder to the PLC on a packaging machine. The cylinder is supplied air through an air pressure sensor and to the pneumatic value for controlling the air supply to the cylinder. As the name says, the cylinder acts in two ways by getting supplied air through inlet airflow sockets on each side of the cylinder.

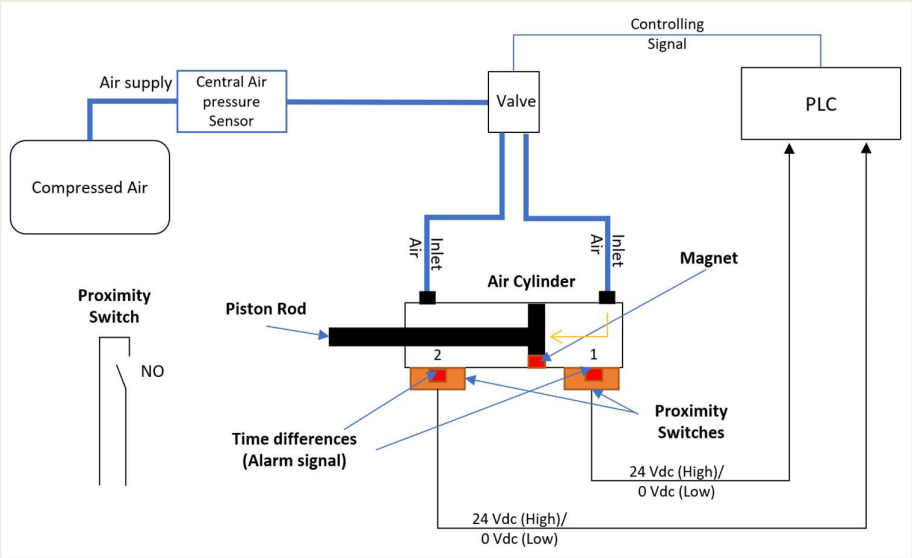


Figure 2.5: System sketch for signal process of piston rod double acting cylinder, proximity switches and PLC.

At the bottom of the cylinder, two proximity switches are mounted. The switches contain a magnet that triggers a normally open (NO) contact set when it comes into contact with the

magnet mounted on the bottom of the piston rod and sends a 24 Vdc (high) signal to the PLC, indicating cylinder stroke has been enabled. As the piston rod moves away from the switch magnet, the contact set is opened, and then the PLC is signaled with a 0Vdc (low) signal. This process is repeated for the switch at the opposite end of the piston rod. Within the PLC programming blocks, there is a function block for each cylinder that counts the number of strokes and calculates the distance of the cylinder's operation. The PLC counts one cylinder stroke when it receives a high signal from switch (1) and a low signal from switch (2). Similarly, when switch (1) is low and switch (2) is high, it will read another stroke from the cylinder. The total operating distance of an air cylinder with a stroke length of 200 mm is calculated by multiplying the stroke length with the number of strokes of the cylinder.

The time difference between the switches, or, in other words, the time the piston rod takes to move from switch (1) to switch (2), is used to indicate an alarm. The normal time for the cylinder stroke is known, and if the piston rod uses a longer time during the stroke, then the PLC generates an alarm. Typically, this occurs when either something blocks the cylinder or the result of wear and tears on the cylinder.

2.3 Data Collector

The data collector shown in Figure 2.1 is the heart of the application. It should collect data from the PLC and other sensors that could directly be connected to the data collector. In addition, it should store the collected data within the local database and present data in a data visualization tool. Based on the requirements for this application, it was decided during the analysis for the application carried out in the project fall of 2021 that WizX, illustrated in **Error! Reference source not found.**, will be used as a data collector system [2]. WizX is an Industrial Internet of things (IIoT) Gateway developed by Goodtech in response to the needs of existing customers. It is designed to measure different types of sensors and read data from PLCs. See Appendix B for more detailed specifications. WizX uses open-source software to build its software platform (WizX Core). In the last few years, different industrial solutions have been developed using WizX, such as automatic collection and reporting of critical data for Glasopor's products' quality [5].

2.3.1 WizX Software Components

WizX Core software is comprised of several components. This section aims to provide an overview and description of the WizX Core software components.

2.3.1.1 Docker

Goodtech uses Docker to create and publish its WizX Core software so that it can be easily managed and accessed by other developers within Coodtech. The Docker platform, which is a free, open-source containerization platform, allows developers to create, publish, and manage applications in lightweight virtualized environments, known as containers [6]. Containers comprise one of the main components of the Docker infrastructure. It allows developers to assemble an application, including its components, such as libraries and dependencies, into a single package. In order to get a better understanding of Docker, Figure 2.6 illustrates the workflow of Docker with its components [7]. When creating an application, the developer defines the application's dependencies and requirements in a file called [2]Dockerfile. This Dockerfile is used to create a Docker image [8]. Docker Image contains all of the information necessary to run an application, and when the Docker Image is executed, the Docker Container is created. In other words, containers are runtime instances of Docker Images. These images can be accessed through a cloud repository called Docker Hub or Docker Registry [6]. When the Docker Images have been built and uploaded to Docker Hub, other developers can pull them and use them to build Docker containers. In this way, other developers will always have the latest version of Docker images where they can pull and create a container. Thus, Goodtech has chosen the Docker platform for running WizX Core, which has better version control of its software. When it comes to security, since all the containers are isolated from each other, this makes it secure in case there is a cyberattack on the

[2] Dockerfile – describes the steps necessery to create a Docker Image.

System Description

Docker is characterized by many advantages, including speed, portability, and scalability [8]. In addition, because the Docker Containers are lightweight, development, testing, and deployment of applications can be done faster.

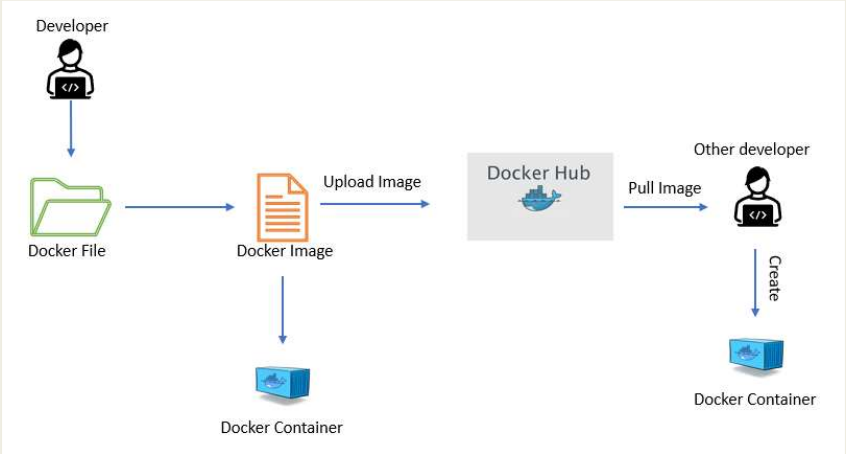


Figure 2.6: Docker componets and workflow [7].

WizX Core components consist of seven separated and isolated container-based images; Node-RED, Grafana, OPC UA Server, MQTT Broker (Mosquitto), TimescaleDB, Portainer Templates, and Portainer. These images are stored in the Docker Hub to be accessible to other users. MQTT and OPC UA are used as communication protocols between those containers. Docker doesn't have any Graphical User Interface (GUI), which may be considered a disadvantage of using Doker as a software development platform. Thus, to visualize and facilitate the management of images and containers within the WizX Core, an open-source management user interface (UI) Docker called Portainer is used [9]. Figure 2.7 shows the Portainer dashboard, which shows the number of containers, networks, volumes, stacks, and images on the WizX core.

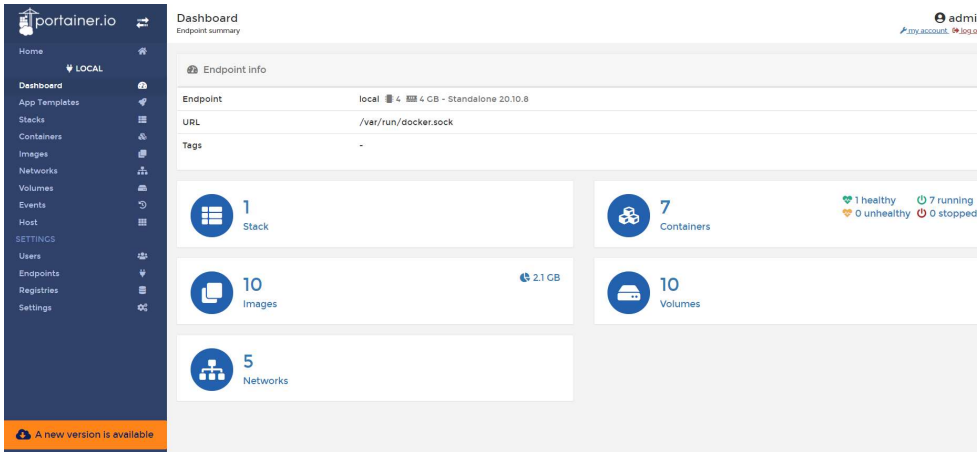


Figure 2.7: Portainer UI for the WizX Core [2].

When WizX hardware is powered, these containers will start automatically and can be monitored from the Portainer after a few minutes, depending on the network's quality. One can start, stop, remove or add a new container from the Portainer UI.

2.3.1.2 Node-RED

WizX Core uses Node-RED as a programming tool to read data from devices in the field and runs as an isolated Docker container under the Docker platform [2]. Node-RED is a free, open-source logic engine developed and maintained by IBM that allows programmers of all levels to interconnect physical I/O devices, cloud-based systems, databases, and APIs. It is built using [3]Node.js, a JavaScript framework that provides a lightweight development environment and runtime environment [10].

Node-RED is a browser-based [4]flow editor which enables developers to wire together flows using the wide range of nodes within the Node-RED library. The flow editor layout has three main areas; Node Panel, Sheets Panel, and Info and Debug Panel, as shown in Figure 2.8. The Node Panel contains a list of nodes which is categorized into types of nodes that can be dragged and dropped into the Sheet Panel. In the Sheet Panel, flows are created by placing nodes in the panel and connecting them. In the Info and Debug Panel, there are tabs for “info” and “debug.” The info tab gives users information about the node's properties when selected within the Sheet Panel. The “debug” tab displays the message passed to the debug node and the log message from the runtime.

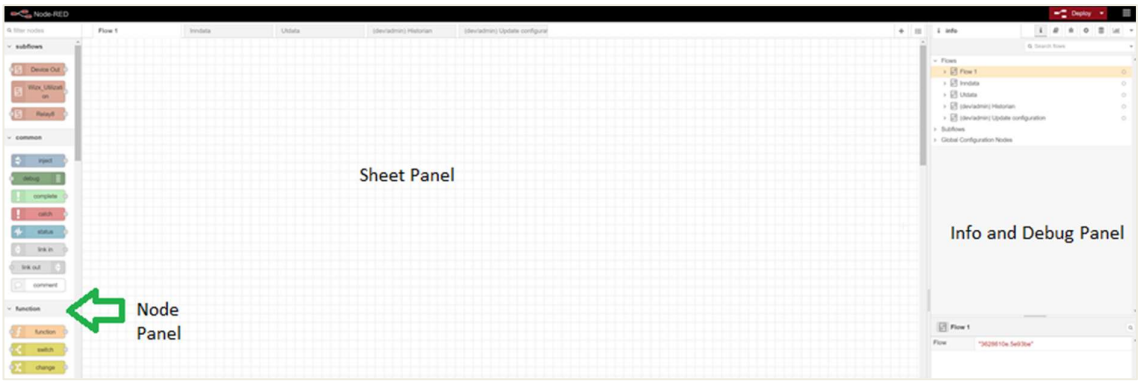


Figure 2.8: Node RED flow editor layout shows the three main areas [2].

The flows are created and managed by different types of nodes, where each node serves a specific purpose; it receives some data, does something with it, and then forwards the data to the next node or completes processing the data. The nodes mainly fall into three different category types; Input nodes (e.g., Inject node), Output nodes (e.g., Debug node), and Process

[3] Node.js is an open source JavaScript runtime environment. The tool is built on Chrome's V8 JavaScript engine, which parses and executes JavaScript scripts. It uses an event-driven, non-blocking I/O model, which makes it both fast and lightweight [10].

[4] The flow editor of Node-RED is accessible from any web browser. If the browser is used on the same computer running Node-RED, the Node-RED server can be accessed by accessing the IP address 127.0.0.1 followed by port 1880, which is the Node-RED server's port number. If using a browser on another computer, you will need to use the IP address of the computer running Node-RED followed by the 1880 port.

nodes (e.g., Function node). For example, those three nodes are connected, creating a simple flow shown in Figure 2.9.

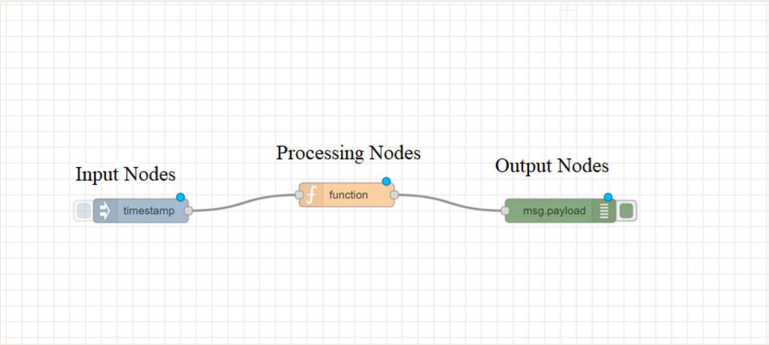


Figure 2.9: The three types of nodes; Input Nodes, Processing Nodes and Output Nodes [2].

A flow can contain many nodes and subflows. Subflows are a collection of nodes that are collapsed down into a single node that can be given a name within the workplace; for example, Figure 2.10 shows a subflow node that generates random numbers. They are used to reduce some visual complexity of the flow to package up a group of nodes as reusable flow used in multiple spaces both in the same or another flow [11].

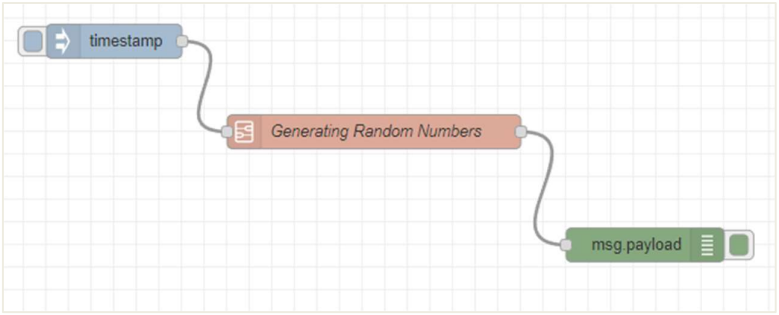


Figure 2.10: An example of created subflow node in Node-RED

When Node-RED is installed, it has several built-in libraries with nodes, but one can also install third-party library nodes. As part of WizX Core, various libraries are included as well as Goodtech-developed libraries that manage, configure, and store data from the devices in the local database. Figure 2.11 shows the Goodtech library module with its node in the Node Panel.

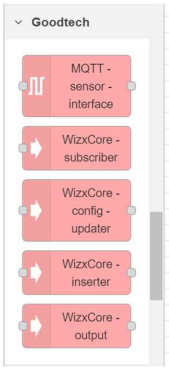


Figure 2.11: Goodtech developed library node.

Another advantage of using Node-RED is that it is also integrated with a dashboard library module consisting of a set of nodes that can quickly provide a live data dashboard instead of implementing another software for data presentations [12]. The created Node-RED dashboard can be presented using a Node-RED UI interface and can be accessed via a [5]web browser

2.3.1.3 Local Database

WizX Core software uses a time-series database with a combination of SQL (Structured Query Language) to store data locally. TimescaleDB is used and run separately in a Docker container. It is an open-source relational database for time-series data. In 2018, TimescaleDB was named the top Database Management System of the year by DB-Engines [13]. Timescale DB is an add-on to PostgreSQL is used, which is designed to make SQL scalable for time-series data. To execute queries and conduct database tasks, pgAdmin 4 is used. pgAdmin 4 is an open-source GUI development platform for PostgresSQL [14].

In Figure 2.12, the Entity-Relationship (ER) diagram illustrates the relationship between the three tables, with a primary key and a foreign key for each. The three tables within this local database are taghistory_data, taghistory_datatype, and taghistory_te. The taghistory_data table contains all data values. In the table taghistory_datatype, a numerical value is assigned to each type of data. The taghistory_te contains data that has an OPC ID, tag names, and history enabled. History enabled is a configuration option for data (metrics) of a device, e.g., a sensor to be presented in a dashboard developed by a visualization tool.

The taghistory_data table stores historical data for only one week. This is accomplished by creating a policy for data retention that is older than one week. The SQL syntax can be found in Appendix C. After one week, new data overwrite the stored data.

The SQL syntax used to create a hypertable for the taghistory_data table and column t_stamp to make storing and querying time-series data more efficient and efficient can be found in Appendix C.

[5] Node-RED UI can be acssed via web browser using the URL that is based on the Node-RED httpRoot pash with/ui added.

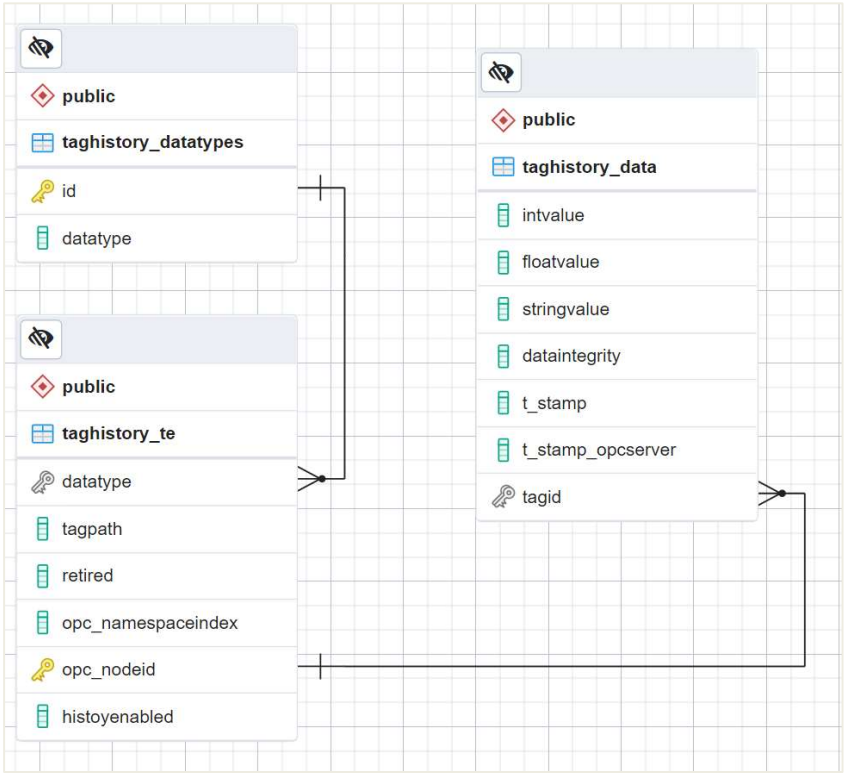


Figure 2.12: ER diagrm of WizX Core local database using pgAdmin 4 Showing the taghistory_datatypes, taghistory_te and taghistory_data [2].

2.3.1.4 Communication protocols and OPC UA

WizX Core supports different sensor interfaces such as I2C, One Wire, SPI, 4-20mA, 0-10V, and Wifi, and communication protocols such as MQTT, OPC UA, ProfiNet, Ethernet, 4G, Bluetooth Low Energy are used. Communication between the Docker Container, MQTT, and OPC UA is used as communication protocols. In this master's thises report, the focus will be on the MQTT protocol and OPC UA that is used to transmit data within the WizX Core. The MQTT and OPC UA are used within the OSI model for handling layer 7, which is the application layer [15].

MQTT

Message Queueing Telemetry Transport (MQTT) is a communication protocol that is very much used in Industrial IoT nowadays [2]. It can be implemented in many programming environments and different platforms such as Windows, Mac OS, and Linux. Because it is a [6]lightweight protocol, it requires minimal resources, so it is also used on small microcontrollers and Raspberry Pi [16]. In the MQTT architecture, there are MQTT Clients and an MQTT Broker. The purpose is to publish/subscribe to data (send/read) between MQTT Clients throughout the MQTT Broker over a network. Figure 2.13 illustrates an example architecture of an MQTT communication protocol with three MQTT Clients; PLC 1, HMI, PLC 2, and an MQTT Broker. PLC 1, as a client, sends temperature data assigned with a specific topic to the MQTT Broker. The broker then publishes the temperature data to the HMI and PLC 2 clients that are subscribers, subscribing to the broker with the same topics.

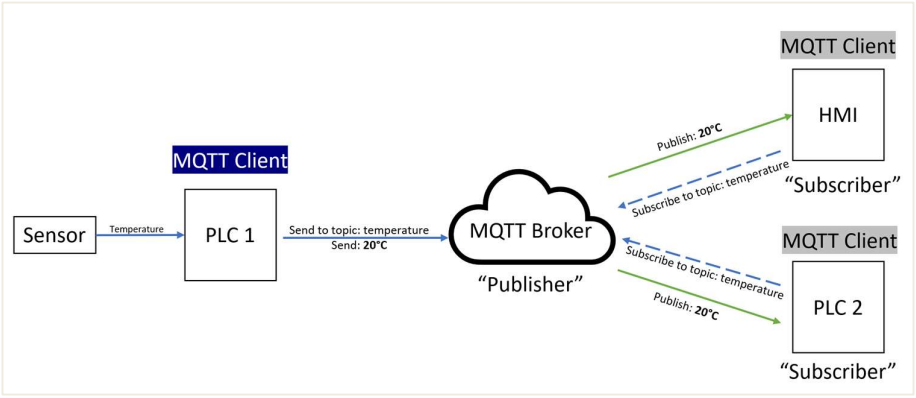


Figure 2.13: MQTT Architecture example.

WizX Core uses the MQTT Sparkplug B Specification protocol and the Mosquitto as an MQTT broker [17]. Mosquitto is a free and open-source MQTT messaging service and supports the Sparkplug B Specification protocol [18]. The Mosquitto MQTT (Server) runs in a separated Docker container and is connected to the Node-RED Container using a library module called node-red-contrib-sparkplug that provides the connection to the Moquitto broker server [19].

[6] MQTT is considered a lightweight protocol because its messages are relatively small in code size. Every message contains four components: a fixed header containing two bytes, an optional variable header, a message payload containing 256 megabytes (MB), and a quality of service level [16].

System Description

OPC UA

Open Platform Communication Unified Architecture (OPC UA) is a machine-to-machine or computer application-to-machines communication protocol for industrial automation developed by the OPC Foundation, which is the next generation of OPC technology [20]. OPC UA communication architecture is comprised of OPC UA Clients and OPC UA Server components [21]. In contrast to the classic OPC DA, OPC UA provides more open communication channels, better security, and a more comprehensive information model. Within WizX Core, the OPC UA communication protocol is used and integrated into Node-RED by installing the Node-RED library module node-red-contrib-opcua [22]. Both the UPC UA Clients and Server are implemented in the Node-RED flows. The OPC UA Clients are used to read/write/subscribe/browse the OPC UA Server. The OPC UA Server runs as an isolated Docker Container within WizX Core and is integrated with the MQTT-Sparkplug interface that can work as MQTT Client and subscribe to the collected data in the MQTT Broker (Mosquitto).

2.3.1.5 Data Visualization Tool

For data visualization, WizX Core uses Grafana and runs as a separated Docker Container [23]. Grafana is an open-source query and visualization tool that runs in a web browser and can be accessed on a mobile device. Grafana allows developers to query external data sources through its query tool. Data source plugins must be installed and configured for Grafana to retrieve and convert data from a data source and visualize it in the panels[24]. Within the WizX Core Grafana container, the PostgreSQL data source plugin is installed and used. Once data are retrieved, they can be visualized and presented within different panels. Grafana offers various panels, including the graph/time-series panel, stats panel, gauge panel, and table panel. Each panel includes a query editor for querying metrics from data sources.

In addition to monitoring metrics, Grafana also allows users to create alerts that send out notifications, such as emails, when metric thresholds are crossed. With the latest version of Grafana, alerts can only be configured within the graph or time-series panel. Users can create alerts based on different conditions for the retrieved metrics. Once the retrieved metrics match the altering rules, an alert is displayed on the panel, and a notification is automatically sent [23].

Figure 2.14 illustrates an example of WizX Core model architecture in which the WizX (hardware) is a collector for collecting metrics from both a PLC and a sensor. When the metrics are stored in TimescaleDB, they can then be queried using SQL within the PostgreSQL data source plugin. PostgreSQL data source plugins format data and present them in the panel presented in a dashboard. The created alert rules are executed in the graph or time-series panel if readings from the sensor or PLC go above a certain threshold. An alert is then visualized on the panel, and an email notification is sent.

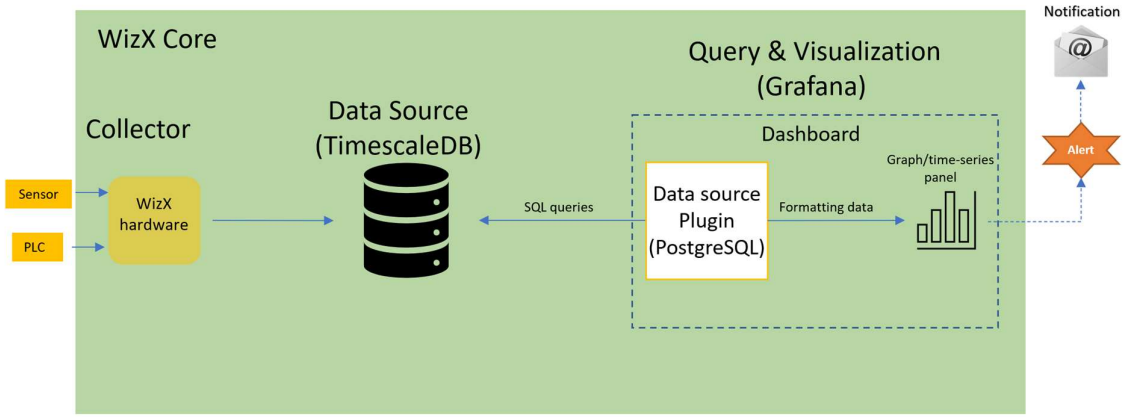


Figure 2.14: A example of WizX Core artitecture viwing metrics from a PLC and a Sensor in graph/time-series panel within a dashbord using Grafana.

2.3.2 WizX Core Structure and Data Flow

The default WizX Core contains three Node-RED flows; In Data, [7](admin) wizxcore, and Out Data. MQTT protocol is used for exchanging messages between the OPC UA Server as MQTT client and MQTT Broker (Mosquitto). The block diagram in Figure 2.15 illustrates data transmission from two devices (PLC and sensor) to the visualization tool (Grafana). The diagram shows that a specific device node for the PLC and sensor from the Node-RED library is used for reading data and configuring them with the device id (tag name) and data type. The device data are then prepared and formatted to an MQTT Sparkplug B object and collected into a single MQTT Sparkplug B Device/Node definition using a sub-flow, which becomes one MQTT Sparkplug Node that contains two devices. Form the this MQTT Sparkplug Node, metrics (tag name, value, and data type) and timestamp from the devices are sent to the MQTT Broker (Mosquitto). The collected data are then subscribed by the OPC UA Server that is integrated with the [8]MQTT Sparkplug interface. The broker publishes all the data that the OPC UA Server is subscribed to. In this way, all the device metrics that are collected in the broker will also be presented in the OPC UA Server. Within the OPC UA server, live data from the devices can be monitored using UaExpert software as OPC Client. From the (admin) wizxcore flow, the OPC UA Client scans the OPC UA Server using the browse function for device metrics automatically or manually once a new device has added or changed its data type or tag name. The wizxCore Config updater node gets an array of inputs from the OPC UA browser and updates the taghistory_te table within TimescaleDB. The updates may consist of new metrics, changed data types for metrics, when metrics get deleted from the UA server, or history enabled has changed. After updating the database, the next step is to inform the Historian part of the flow to subscribe to updated metrics in the database. This is done in parallel using the

[7] (admin) wizxcore is the name of the Node-RED flow.

[8] The OPC UA Server is used within WizX Core is integrated with MQTT Sparkplug interface which has the ability to subscribe to the MQTT Broker as client.

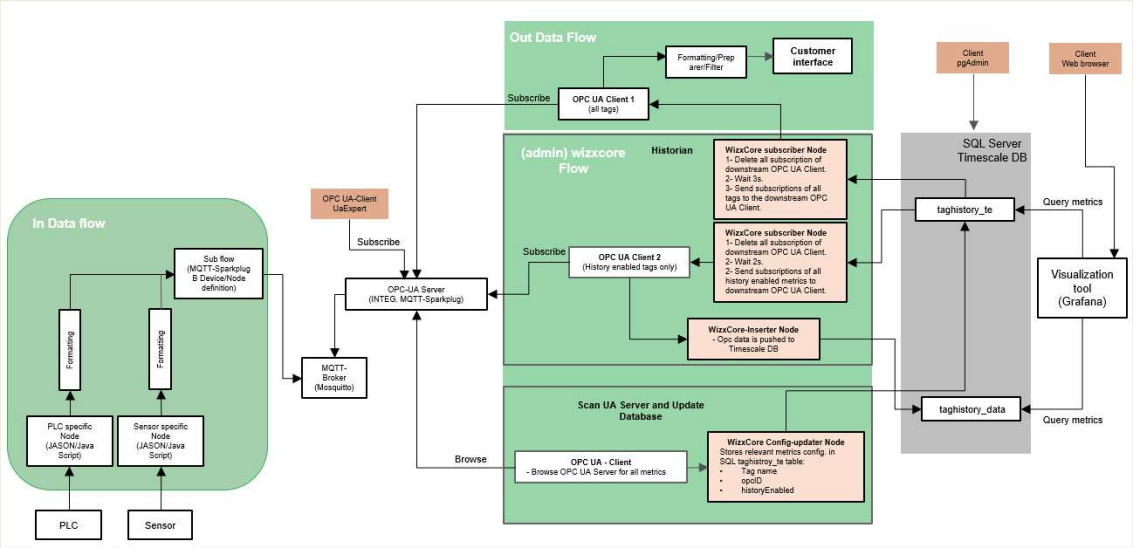


Figure 2.15: WizX Core structure and data flow diagram.

Goodtech developed WizxCore-Subscriber nodes connected to OPC UA Clients 1 and 2. The WizxCore-Subscriber node queries the taghistory_te table and then first sends a message to the OPC UA Client 1 in the Out Data flow to delete all metrics that it has subscribed to before, waits for three seconds, and sends a new command to subscribe to all metrics on the OPC UA Server. The data are then prepared/formatted/filtered before sending them to the customer delivery interface block, e.g., a cloud-based database. At the same time, another WizxCore-Subscriber node checks the database table for which device metrics are history enabled, meaning data will be presented in the Visualization tool (Grafana). When it gets those metrics that are history enabled, it sends a message to OPC UA Client 2 to delete all the metrics, wait for three seconds, and then it sends a new message to subscribe to all history enabled metrics on the OPC UA Server. The client then sends those metrics to the WizxCore-Inserter node, injecting them into the table taghistory_data within the local database TimescaleDB. The data visualization tool (Grafana) queries both tables with metrics and timestamps using SQL syntax and displays them in the dashboard using different, e.g., a graph panel.

2.3.3 WizX Core Components

As described in the 2.3.1.1 subsection, WizX Core is comprised of seven isolated Docker Containers [2]. These containers represent individual software components within the WizX Core. An overview of all WizX Core components can be found in Figure 2.16. When WizX (hardware) powers up, all the containers will start up and run after 2-3 minutes. As illustrated in the Figure 2.16 diagram, the Node-RED container handles the device integration and sends data devices from the PLC and sensors with a namespace topic to the MQTT Broker container (Mosquitto). OPC UA Server container subscribes to the same topic. Node-RED flow (admin) wizxcore from the Node-RED container browses the OPC UA Server for device metrics and updates the local time-series database TimescaleDB container using the OPC UA Client. At the same time, the data from the updated database is sent to Out Data flow, where it is being prepared and can be sent to the customer.

System Description

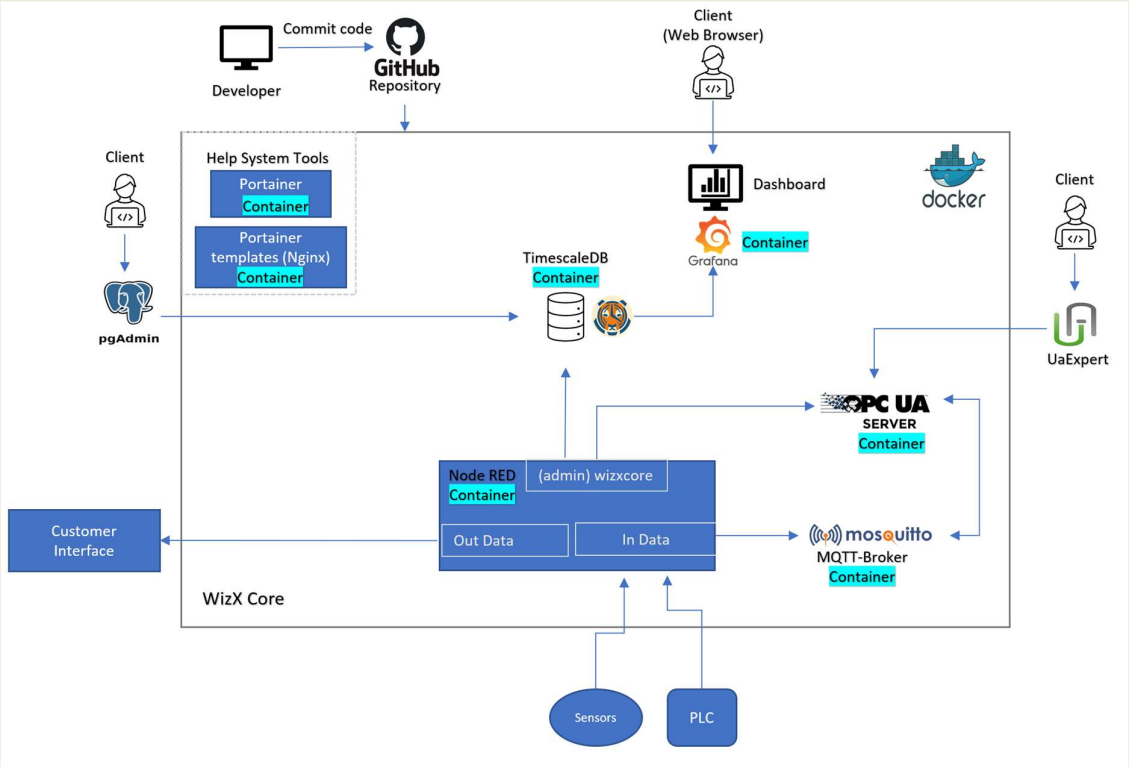


Figure 2.16: An overview of WizX Core components [2].

Portainer container and Portainer template container are components used in the system helping tool. GitHub is used for version control of WizX applications, where each application has its own repository with all files, and their version histories are saved.

Since WizX is a container-based platform, each container can be operated independently, allowing applications to be more flexible and function as a distributed system.

As seen in the Figure 2.16 diagram, pgAdmin 4 is used to manage the database and monitor the data flow within the local database. Also, UaExpert software as OPC UA Client is used to monitor live data from OPC UA Server. These tools can be useful when troubleshooting and testing the system.

2.4 Introduction to Application Packaging Machine

The Application PM will be an extension of the WizX Core software, which aims to monitor one or several packaging machines by collecting the distance of the cylinder stroke from the air cylinders, VFDs running hours via the PLC, and data from any other sensors. Figure 2.17 shows the system sketch for the application. It is an extension of the system sketch presented in Figure 2.1 and will be implemented in this master's thesis. The collected data should be presented in the dashboard by using the Grafana visualization tool. The dashboards will be integrated with alerts and email notifications, whereby the approximate life expectancy distance of the air cylinders provided by the supplier will be used as the threshold value within

the alert rule condition for each cylinder. When the collected stroke distance from a cylinder is near its operating life expectancy, the operator is notified by an alert on the dashboard and receives an email notification. The running hours of the VFDs will be monitored, including email notifications when the servo motors are approaching operating hours that require maintenance. In addition, the application should be flexible and able to gather data from any other sensors that may be added to the system.

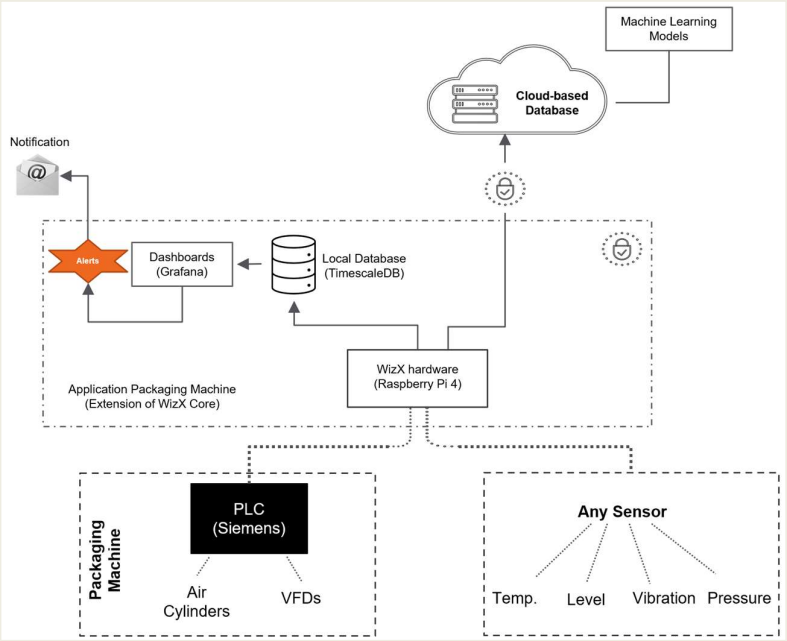


Figure 2.17: Application Packaging Machine system sketch.

The collected data will be stored in the cloud-based database, which can be used as historical data in developing ML models that can predict the wear and tear of the air cylinders on the packaging machine. The implementation of ML is not emphasized in this master's thesis; rather, a proposal on how ML can be used to predict the maintenance of packing machines in order to ensure maximum operating efficiency is provided.

The Application PM will be offered to customers as a service in conjunction with the packaging machines and will be controlled and operated by Goodtech AS in the Moss department.

3 Methods

This chapter describes the working methods used for the master's thesis. In general, software development processes are defined along with what kind of software process model Goodtech follows and how this model compares to the Unified Process Model. ML and its methods and algorithms are described briefly as to how they can be used within the Application PM for the predictive maintenance of packaging machines.

3.1 Software Development Process

Software Development Lifecycle (SDLC)

A software process is a set of related activities that leads to software production. These activities may involve the development of new software or a modification of existing software. Developing any software will have a lifecycle that includes phases during the development. The phases may consist of planning, Requirements Analysis, Design, Implementation, Testing, Deployment, and Maintenance, as illustrated in Figure 3.1 [25].

During the planning phase, developers gather to discuss upcoming projects [26]. The project schedule is an imperative step in the planning phase. The requirements are collected using the project description, emails, and meetings with the customer in the requirements and analysis phases. For collecting requirements, different methods are used. One of them is the FURPS+ model, an acronym for Functionality, Usability, Reliability, Performance, and Supportability [27]. The collected requirements are then analyzed to describe how the system should work. Before the next step is taken, the development team could outline the whole system process and discuss whether the system will meet the user requirements or not. If it does, then the design phase of the software based on the collected and analyzed requirements will be carried out.

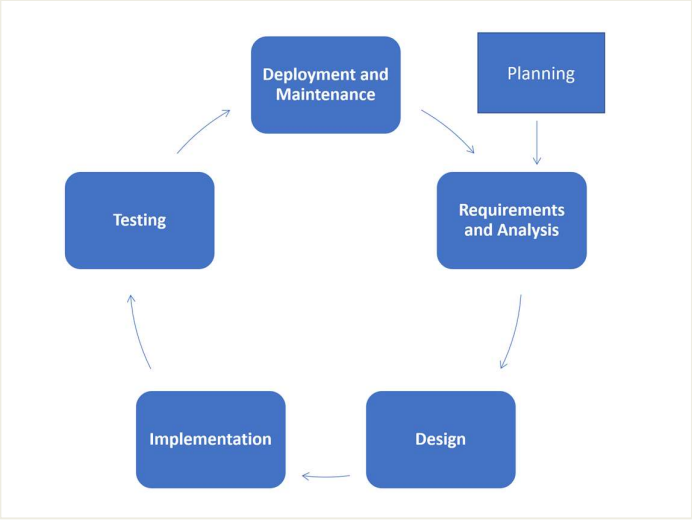


Figure 3.1: The Software Development Lifecycle [25].

Methods

During this phase, an outline of the entire system is created, which defines the overall system architecture. The next phase is the implementation phase, also known as the coding phase. Usually, the working process is divided into different modules, and the coding is performed within each one. A testing phase follows the implementation phase, in which the code is tested to determine whether it meets the requirements of the end-user and if it is working as intended. There are four types of testing: functional, non-functional, structural, and change-related testing. Each type of testing has its own objectives, advantages, and procedures that need to be documented.

The software can be delivered/ deployed to the end-user following a successful testing phase. To maintain the software, a team will be appointed which is capable of providing timely updates and alerts related to them.

Common problems

When developing software, certain problems are common. The lack of proper specifications and poor requirements will lead to poor development of the design, resulting in an incomplete software product. Team miscommunication is another common problem. When teams fail to communicate effectively, delays can occur, which will affect the planned schedule. For this to be prevented, it is essential to have a well-defined approach to project management as well as regular meetings involving all project disciplines. It may be the case that sometimes a client presents unrealistic features in the software request that is not feasible during the review of requirements. Thus, it is very important to understand what the client wants and analyze the requirements thoroughly before proceeding to the next phase. An unrealistic schedule is another issue one could encounter, where a client expects a 12-month project to be completed in three months. Finally, testing is also likely to be inadequate if the test team does not understand the application's business, therefore failing to generate good test cases.

Software Development Process Models

There are several phases involved in software development. It starts with an idea, business requirements, or a concept to build something into a software product. It is important to choose the right model. Various models have been developed and used throughout the years, each representing a process from a unique perspective. This is why understanding each model is important for choosing the most suitable model for a project. Among the most commonly used models are described briefly below:

Waterfall Model: This is a basic model of software development that is simple to understand. The requirements must be clearly stated from the beginning. The model is suitable for small projects [28].

Iterative Model: Adaptable to changing requirements. The development of software is incremental. Iteration is easy to manage because the entire project is broken down into smaller project [29].

Spiral Model: It is a combination of waterfall and iterative development. Because it is a complex model, it is suitable for large projects. Risk analysis and management are integrated with the model. With each iteration, one analysis mitigates the risk [30].

Methods

V-model: This model extends the Waterfall model, which includes additional testing phases compared to the Waterfall model. Like the Waterfall model, the V-model requires complete requirements at the start of the project. It is suitable for both small and large projects [30].

3.2 Software Development Process Model within Goodtech

Goodtech AS does not follow a specific software development process model rather, the model they have used is project-based, consisting of a project manager and a team of developers. Tasks within a project are delegated to different developers and are divided into iterations where each iteration represents the whole development process; *the requirements phase*, *analysis & design phase*, *development phase*, *testing phase*, and *deployment phase*. Figure 3.2 illustrates the process model, which is more and less similar to the Iterative Model. The model begins with preparing requirements based on an idea or a system description. In the analysis and design phase, the requirements are analyzed and designed. In the development phase, developers begin coding based on the results of the previous phase.

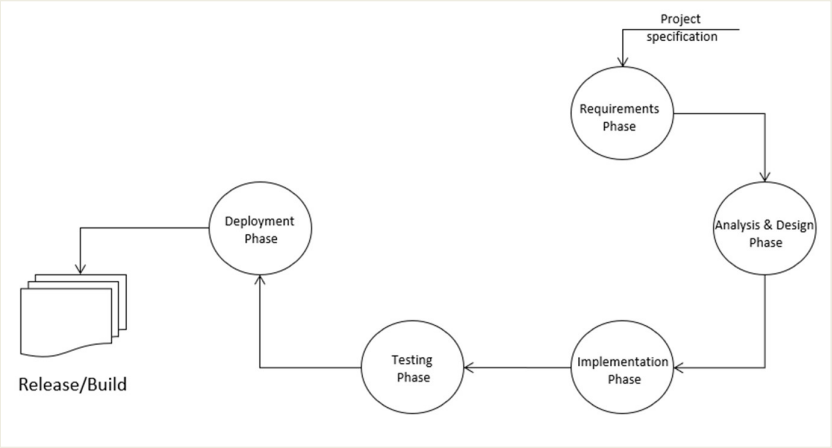


Figure 3.2: Software development process within Goodtech AS.

Once all the identified requirements have been developed, the next phase will be testing. During the testing phase, the requirements are reviewed to ensure that the code that has been developed meets the requirements. If all the testing is completed smoothly, the next step is deployment. In this phase, certain features of the software will be released. The process then repeats itself during each phase of the software development process until the development of the software is completed. Like any other development process model, this model also has pros and cons. The following are the advantages and disadvantages of this development process model:

Advantages

- Complete requirements are not required at the beginning of the development process. Requirements can be modified, added, and removed as one goes through the iterations. This is a big plus compared to the Waterfall Model.
- A small portion is developed in each iteration, which is easy to develop and test.

Methods

Disadvantages

- It's difficult to architecture a system with incomplete requirements. It depends on the technologies that are used. Sometimes the change or new requirements observed in the system may not be supported by the tools selected for the development process, such as the programming language, database, or intermediate software for communication services.

It is decided to follow this development process that Goodtech AS has used for developing the packaging machine application.

Compare the Goodtech model with the Unified Process Model (UP)

The Unified Process (UP) has emerged as a popular software development method for developing object-oriented systems using four phases and nine workflows [31]. The four phases are Inception, Elaboration, Construction, and Transition. The workflows consist of six core flows; business modeling, requirements, analysis & design, implementation, test, deployment, and support workflow; configuration and change management, project management, and environment. Figure 3.3 shows the diagram of the UP process. The phases are placed at the top, and on the left-hand side are the workflows. As illustrated in the figure, each workflow except deployment goes through all the phases. This means, for example, in the inception phase, one does a little bit of every workflow. Still, as you can also see, there is more weight to the business modeling and requirements in the inception phase. In the elaboration, construction, and transition phases, there are iterations. The UP model is mostly used for large development projects.

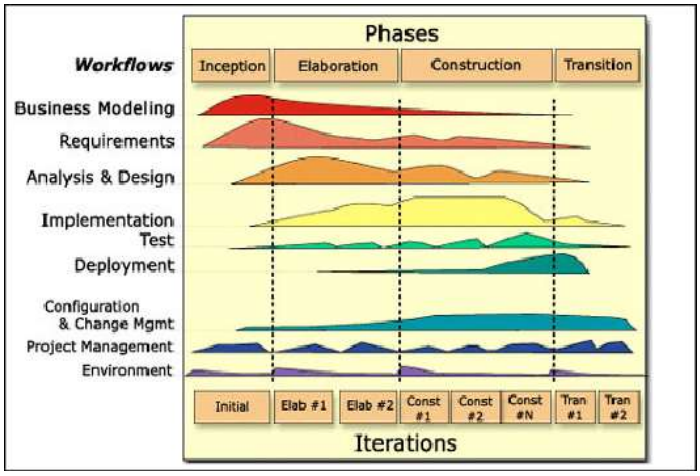


Figure 3.3: Unified Process [31].

Compare the Goodtech model with UP Model

Considering both the UP model and the Goodtech Model, both are iterative-based and allow customization of the process in a different manner. The development of UP is based on a series of short, fixed-length small projects. Each project's outcome might be integrated, tested, and executed as a whole. Every small project involves its own requirements analysis, design, implementation, and testing activities. A Goodtech model consists of multiple iterations aimed

at continuously improving a system. It is done through the use of cyclic feedback and adaptation as central drivers for developing a suitable system where it evolves incrementally over time. Unlike the UP model, the Goodtech model is suitable for both small and large projects. The UP model is too complex for small projects, even though one may argue that it can be used for small projects. The developer must have a deep understanding of the UP model.

3.3 Machine Learning (ML)

In the prior analysis, the definition of Machine Learning (ML) was included [2]. A brief discussion of the relationship and differences between Machine Learning, Deep Learning, and Artificial Intelligence is presented in this subchapter. It also describes the types of ML and their methods. Moreover, it presents a data flow that will be used for the further development of the application with the implementation of Machine Learning models that are used based on the historical and live data in the cloud so that it can give better advice to operators and status regarding air cylinder maintenance on packaging machines. How should the history dataset be preprocessed, what is the best algorithm to use, and how should it be implemented?

3.3.1 Relationship and differences between Machine Learning, Deep Learning, and Artificial Intelligence

Nowadays, companies and developers worldwide talk about Artificial Intelligence (AI), ML, and Deep Learning (DL). There is a common misuse of the terms AI, ML, and DL when referring to technology. To better comprehend the significance of these acronyms, it is necessary to understand that they are all fundamental components of IA [32]. As illustrated in Figure 3.4, AI is a technique that allows computers to replicate human behaviors. The subset of AI we have is ML. ML can be defined as a subset of AI techniques that use static methods to enable machines to improve with experience. Under ML, there is another subset which is DL. DL is ML, which makes the computation of multilayer Neuron Networks feasible. The difference between ML and DL is that ML requires algorithms that learn patterns in datasets and then apply them to decision making, whereas DL can learn by processing its data and is quite similar to the human brain, which identifies, analyzes, and makes a decision.

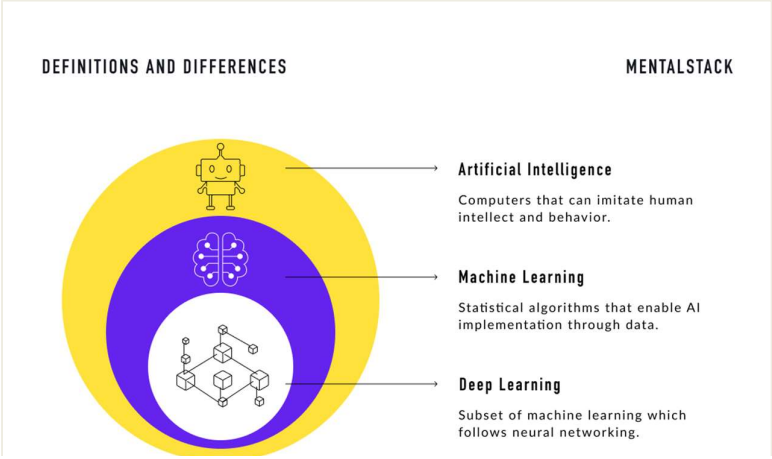


Figure 3.4: Relationship and differences between AL, ML and DL [32].

3.3.2 Machine Learning types and methods

Like humans, machine learning employs a variety of learning techniques. Three distinct types of ML are illustrated in Figure 3.5; *supervised learning*, *unsupervised learning,* and *reinforcement learning*. Supervised learning is task-oriented, unsupervised learning is data-oriented, and reinforcement learning results from mistakes. Different algorithms have been developed throughout the year based on each learning type.

Supervised learning includes Logistic Regression, Classification, Naive Bayes Classifiers, K-NN (k nearest neighbors), decision Trees, and Support Vector Machine (SVM). No one method (algorithm) can be used for all kinds of problems, but before choosing a suitable algorithm, the developer must know the problem well and what kind of outcome one wants to achieve. In supervised learning, the algorithms are used to deal with two types of problems: *regression* and *classification,* as shown in Figure 3.5 [33]. Regression problems are when the output variable is a real value, such as "dollars" or "number" or "weight," while the classification problem is when the output is a category, such as 'Red' or 'Blue' or 'disease' and 'no disease.'

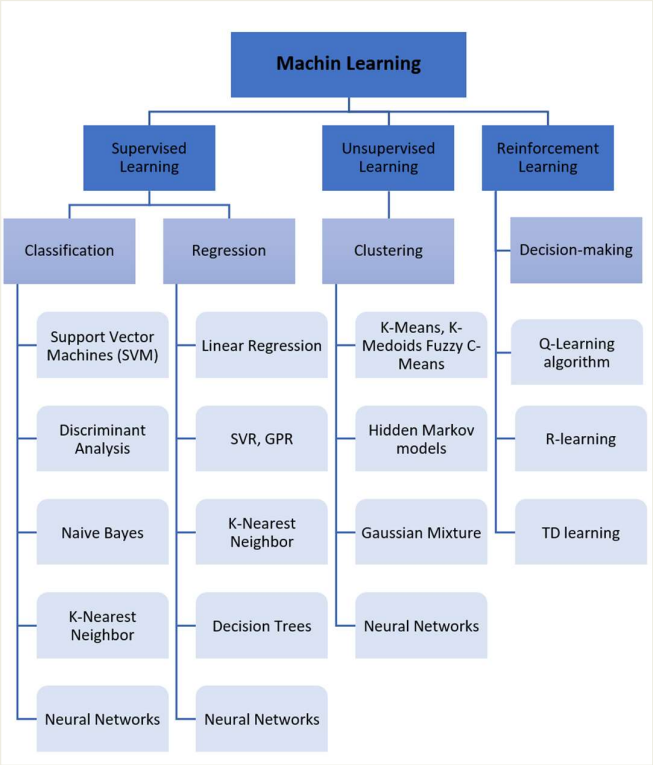


Figure 3.5: ML types and algorithms [33].

Supervised ML helps solve real-world computation problems such as Speech Recognition, Spam Detection, and Object-Recognition.

In unsupervised learning, the problems are less defined than in supervised learning and more complex to solve, but the solution is more powerful if handled well.

Methods

Unsupervised learning problems are classified into two categories of problems: *clustering* and *association*. [34] Clustering is where the aim is to discover the inherent groupings in the dataset, such as grouping customers by purchasing behavior.

Common algorithms for performing clustering include k-means, hierarchical clustering, Gaussian mixture models, hidden Markov models, self-organizing maps, and fuzzy c-means clustering.

Reinforcement ML is a behavioral machine learning model that is similar to supervised learning, but the algorithm isn't trained using sample data. This model learns as it goes by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy for a given problem.[35] This learning model is much used in robotics for industrial automation and data processing [35].

3.3.3 Proposal for implementing ML in Application Packaging Machine

Based on the task description of this master's thesis found in Appendix A, it should propose how the application can be further developed by implementing ML to predict the maintenance of packing machines to ensure maximum operating performance.

Implementing ML algorithms in packaging machines can identify and predict failures within the air cylinders before they occur. This will reduce maintenance expenses, avoid an unplanned shutdown and improve the efficiency of the packaging machines. For this purpose, the stored history dataset produced by the Application PM can be used to train the ML models, which can then be compared with real-time data from the Application PM to predict whether or not the air cylinders need repair or be replaced. Thus, another application should be developed and used to preprocess history data, develop the ML, and make predictions. This application can be called Predictive Maintenance (PdM).

To develop reliable ML algorithms that can identify and pridect breakdowns in the cylinders, more measurement data must be established in the system that gives more information about the cylinder conditions. For this master's thesis, airflow and air pressure sensors have been considered that can be used on packaging machines to collect information regarding the air cylinders' condition. The airflow measures the flow rate of the air that is needed for actuating each air cylinder, while the central air pressure sensor measures the overall air pressure that is needed for operating the cylinders that are operating in the packaging machine. In addition, an alarm signal from air cylinders that are triggered from proximity switches can be retrieved from PLC that gives the status of each cylinder. With a combination of data from those sensors and alarm signals, it is possible to develop models that can provide information regarding the condition of the cylinders.

Various signs can be observed when there is an issue or wear on an air cylinder, such as an air cylinder requiring more air supply and pressure to operate than it would under normal conditions [36]. This issue can be detected by monitoring the central air pressure sensor and airflow sensor for the cylinders. In addition, the same issue can be detected as an alarm that is triggered by the proximity switches from the cylinders, as illustrated in the system sketch in Figure 3.6. The figure shows the signals that can be collected and stored as history train data in the cloud-based database. Two proximity switches are mounted on each cylinder, sending a high or low signal to the PLC depending on whether the cylinder is in operation. As shown in the figure, once the magnet on the cylinder piston contacts the proximity switch, the normally

open (NO) contact closes and sends 24 Vdc, indicating a high signal to the PLC, and when the magnet loses contact with the proximity switch, the normally open (NO) contact opens and sends 0 Vdc indicating a low signal to the PLC. In normal operation, the piston of the cylinder moves from proximity switch 1 to proximity switch 2. The time difference between proximity switches is defined as actuating time in seconds for the regular operation of a cylinder.

As the value of the time difference increases, this indicates that something is blocking the cylinder's movement or something is wrong with the cylinder, then the PLC generates an alarm. The alarm signal can be converted to digital signals of 1 and 0, 1 indicating an alarm and 0 meaning no alarm. These values can be sent to the cloud DB through the Application PM using WizX. In another scenario, if there is air leakage in the system, such as between the valve and the cylinder, the pressure level on the compressor will drop at the same time the airflow value will decrease.

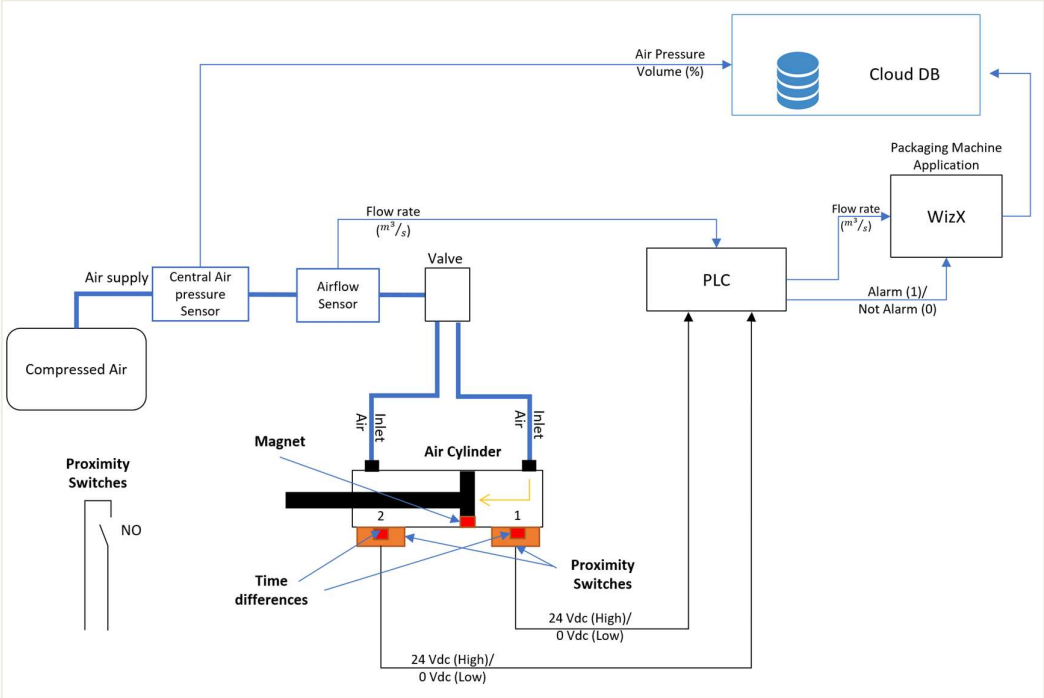


Figure 3.6: System sketch of collecting signal from the packaging machine and store then within cloud database.

At the same time, the time difference between proximity switches will increase. This information will, of course, not mean that there are faults in the cylinders themselves, but rather faults in the system which can cause wear on the cylinders over time. Figure 3.7 illustrates the proposal system sketch for implementing ML that shows the data flow within the system, including the PdM Application and the Application PM. The air pressure sensor and PLC containing the air cylinder, VFD, airflow sensor, and alarm signals from proximity switches are collected into the Application PM using WizX. The collected data are stored in the local DB using TimescaleDB and then queried and visualized in the dashboards using Grafana. At the same time, the collected data are stored in a cloud-based DB using Microsoft Azure Server for MySQL. The stored data in cloud DB for an air pressure sensor, airflow sensor, and alarm data can be exported as a history dataset in a CSV file.

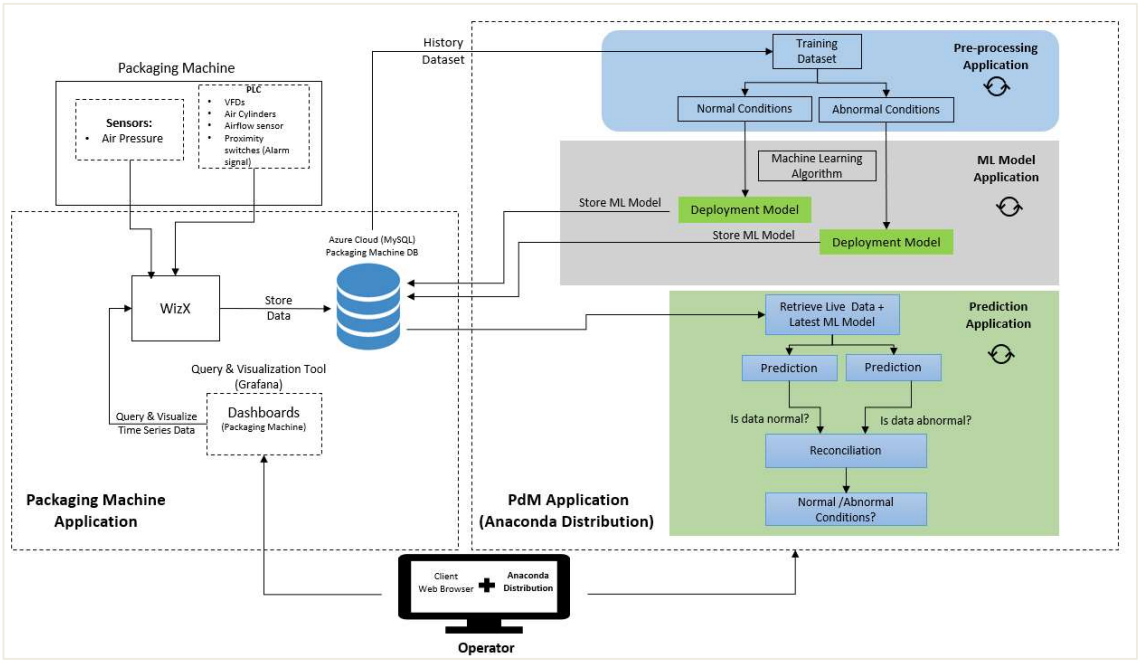


Figure 3.7: System sketch of implemeting ML in packging machine that show the data flow between Packaging Machine Application and PdM Application.

Pre-processing dataset

Any ML model must start with the pre-processing of the dataset. The dataset collected from the Application Packaging Machine is a time-series dataset. These data are simply collected from measurements and downsampled over time. In order to use these collected time series data for the development of ML models, it is crucial that the time series collection span be long enough to cover at least one life cycle of the cylinders. This includes both periods where the cylinders are performing optimally and those when they are beginning to wear out and starts to fail and must be replaced. Once the historical dataset is collected, they have to go through some pre-processing steps and treatment before it can be trained within the ML models. Before doing any pre-processing, it is important to conduct a time series analysis to understand the data by decomposing it into a graphical trend. After understanding the dataset and gaining some insight into it, the preprocessing step can be undertaken.

As illustrated in Figure 3.7, the history dataset is split into the normal conditions dataset, where the air cylinders are operating at their optimal condition, and the abnormal condition dataset, where the cylinders are not operating at their optimal condition. The aim of doing that is to create two ML models where one can be used for predicting the normal condition of the air cylinders and the other one for predicting abnormal conditions of the air cylinders. The operator will then have to reconcile the two model-predicted results. Suppose the certainty of the result of the prediction from the normal condition data is 98% or more, and the certainty of the result of the prediction from the abnormal is 5% or less. In that case, this indicates that the cylinders are in good shape and operate at their optimal condition.

On the other hand, if the certainty of the results from the normal conditions is lower and the certainty for abnormal is higher, then, in this case, the operator has to reconcile between them and maybe raise a warning and create a user note table in the database with time period and information about the fault on the specific cylinder. This information can be useful for defining and splitting the abnormal conditions from the normal conditions dataset. Another way to splitting the dataset, which is easier to obtain, is by detecting and handling outliers in the history dataset. In statistical terms, outliers are data points that do not belong to a certain population. Outliers may also be abnormal observations that are far from the mean [37]. In the case of the central air pressure on the packaging machine, or flow rate for a specific air cylinder, the value may be lower or higher than the average value, which suggests either that the air cylinders may need to be replaced or repaired or that there could be another fault on the packaging machine that must also be noted which the main goal of the ML models. Once all the outliers are detected, they can be handled by storing them as an abnormal condition dataset, and the remaining dataset becomes the normal condition dataset. The following step in pre-processing is to filter the dataset by removing all the missing data within the dataset. There are different techniques for removing the missing data, such as using the technique that removes the entire row or column containing the missing data. Although this technique may not be necessary a useful one because it also provides useful data when removing the entire row or column. A common technique for dealing with missing data for time-series data is to take a mean or moving average, which is taken as the mean of the values within a predefined window, such as a one-month window. The result of this moving average is then used to fill in any gaps in the time-series dataset [38].

The next method for dataset pre-processing is feature scaling. The collected dataset from the air pressure sensor and airflow sensor will not be at the same scale; thus, they have to be scaled. On the other hand, the alarm signal dataset will contain 1 and 0, where the scaling will not be needed. There are different techniques for feature scaling, such as Standardization, Normalization, Min-Max scaling, and Robust scaling [39]. However, the most widely used technique is the Standardization and Normalization. The standardization technique is based on computing the transformed values by computing the difference of each feature value from the mean of the values of that feature and then dividing it by the standard deviation for that feature using Equation (3.1) [40].

$$x_{transformed} = \frac{x - mean\ (x)}{standard\ deviation\ (x)} \tag{3.1}$$

The normalization technique computes transformed values by dividing the minimum value of each of the features by its different values. Then, divide the result by the difference between the minimum and maximum value for that feature using Equation (3.2) [40].

$$x_{transformed} = \frac{x - min\ (x)}{max\ (x) - min\ (x)} \tag{3.2}$$

Now that the history datasets have been pre-processed and separated into two datasets. The final step is to divide the dataset into three sub-sets. A training set consists of 65%, a validation dataset of 20%, and a test dataset of 15%. The total dataset should be divided, and only the training dataset should be used to train the ML models.

Methods

Tools and programming language for developing ML models

ML models can be developed in various programming languages, including Python, R, MATLAB, C#, JavaScript, and C/C++. Python is recommended to be used as a programming language for the PdM Application since the author of this master's thises has used Python for ML model development. Python is also an open-source language that is widely used in the industry or education and has many helpful libraries for simplifying operations, such as NumPy, Pandas Sklearn, Matplotlib, Sicikit-Learn, and Jupyter, which are available as part of the Anaconda Distribution [41]. The Anaconda Distribution is a popular Python and R programming distribution that simplifies package management and deployment. Anaconda features a graphical user interface (ANACONDA.NAVIGATOR) that allows the launching of applications, environments, and libraries.

Selecting the ML algorithm

Based on the system sketch of the proposal setup for implementing ML in the packaging machine application, where a good amount of history dataset must be collected and trained to develop ML models (normal condition and abnormal condition) that can be compared with the live data that is generated from the Packaging Machine Application and predict the air cylinders conditions based on the live data from the central air pressure, airflow and alarm signals from the proximity switches. Thus, the supervised learning method is recommended to use within the PdM Application. Different algorithms within the supervised learning type can be used to solve it as a regression problem, as illustrated in Figure 3.5. K-Nearest Neighbors (KNN) is one of the simplest algorithms that is recommended to be used. KNN uses the 'feature similarity' to predict the values of any new data points [42]. This means that new data (live data) from the air pressure, airflow sensor, and alarm signals from proximity switches are assigned values based on how closely it resembles the points for normal conditions and abnormal conditions in the training dataset. Suppose the new data are close to normal condition datasets with a high percentage and close to abnormal condition datasets with a deficient percentage. In that case, it means that the cylinder is in good condition. In the opposite case scenario, the cylinder condition will fall into an abnormal condition. Thus, the operator must raise a signal and inform the customer. In case of a scenario where new data falls in between both conditions, the operator must reconcile and perhaps send a warning to the customer.

Storing ML Models

As illustrated in the system sketch in Figure 3.7, the trained models associated with date and time should be stored within the same database for Packaging Machine Application in the Microsoft Azure Server for MySQL DB so that an operator could retrieve the latest version anywhere and anytime. A new table within the database must be added where it consists of the model number as integer datatype, model description defined as characters, timestamp, and model file as bytea data type that allows storage of Pickle file. The pickle module is a process that implements a binary protocol for serializing and de-serializing Python object structures [43]. After model development, it can be stored using the "pickling" process for converting the Python objects into a byte stream and retrieving the model again using the "unpickling" process for converting back to Python object structures. To establish the connection between the PdM application and the Azure MySQL Server DB, during the storing and retrieving models, Python library *mysql.connector* must be installed and used [44]. In addition, the pickle Python library must be used for storing and retrieving the ML models.

Running the PdM Application

The PdM Application contains three separated applications; Pre-process Application, ML Model Application, and Prediction Application. The Pre-process Application retrieves the history dataset, pre-processed, and the outcome will be two separated datasets. The separate datasets are then divided into 65% training set, 20% validation set, and 10% test set and exported as CSV files. The ML Model Application trains the training datasets separately for both normal and abnormal datasets. Once the training models have been developed, the models must be evaluated against the validation datasets. As soon as the models are evaluated, the operator will select the best model that has the least error rate and produces a high approximate prediction. This model will then be tested with the test dataset to ensure that it continues to perform well and matches the validation dataset results. The developed ML models are then deployed and stored in the cloud database. The Prediction Application imports the latest stored ML models together with live data from the cloud database and make the predictions. The prediction result from both models can be displayed as a separate graph and numerically. When it comes to how often these sub-applications within the PdM Application should run, the Pre-process Application should run when the cylinders have gone through at least one life cycle. The ML Model Application should run several times at the beginning of each new history dataset. While the Prediction Application should run continuously when the packaging machine is in production mode.

4 Analysis and Design

This chapter covers the analysis and design of the application, which includes a summary of previous analyses and further analysis of the cloud-based databases and the dashboard. In addition, there is a need to design and analyze the security aspects that protect the data within the database and the application, both locally and in the cloud also covered.

4.1 Analysis

The section presents a brief summary of the analysis of the application that was carried out during the Project in the autumn of 2021 [2]. In addition, further analysis of the application that includes the dashboard, cloud-based database, and securing data is also presented.

4.1.1 Summary of the prior analysis

An application requirements document, including collection requirements, was drafted based on the task description for the application. [2] Using the collected requirements, a use case diagram was created. It contains three use cases that illustrate each function of the system and how the application will operate. Further analysis of the uses case diagram, as shown in Figure 4.1, is included in this Master's thesis with an updated description of the use cases and actors. Actors are external components to the system that carries out a specific role by interacting with the use cases.

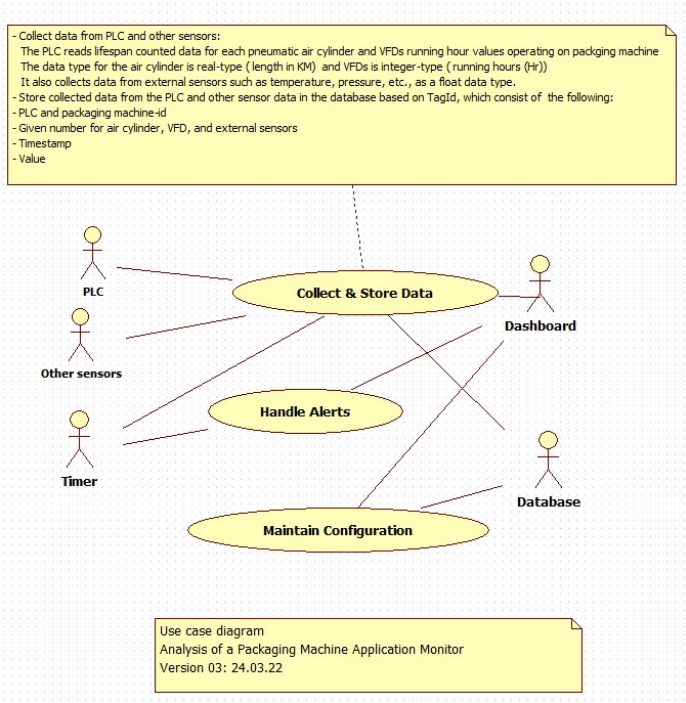


Figure 4.1: Use case diagram for Application Packaging Machine [2].

These use cases are described in more detail as follows:

Collect & Store Data: This use case is responsible for collecting data from the actor *PLC* and *other sensors* based on sampling time represented by actor *Time*. The use case is also responsible for storing data in the cloud-based database and visualizing data on the dashboard.

Handle Alerts: In this use case, alerts are created and configured by creating and configuring rules and conditions for each air cylinder's lifespan and the VFD running hours. These roles and conditions are configured on the dashboard. If the air cylinder lifespan is approaching its service life or if the running hours of a VFD are about to exceed the limit for motor maintenance, the created rules will be violated, and an alert will appear on the dashboard as well as an email being sent to the operator.

Maintain Configuration: It is responsible for maintaining the configuration of the dashboard and cloud-based database. On the dashboard, the configuration includes obtaining the number of packing machines, air cylinders, VFDs, and sensors. It is also responsible for updating/changing the alert limitations via the dashboard and resetting alerts after replacing air cylinders or after resetting the VFD running hours after motor maintenance. Additionally, it handles the configuration of new sensors, air cylinders, packaging machines, and VFDS. A UI is provided by the application that accesses the cloud-based database in order to configure the database in the cloud. Access to the database needs to be secured by a login function with different users. Once logged on, a user can configure the tables in the database by adding new machines, air cylinders, VFDs, and sensors as well as updating and deleting existing ones.

Following the analysis, it was determined that the application PM would be an extension to WizX Core with its software components. As well as a Raspberry Pi 4 with a 32 GB SD card for collecting data from the PLC and other sensors. Raspberry Pi 4 is powered by a power supply for converting 230VAC to 5 VDC and a wireless mesh communication module[45] for connecting wireless sensors, as shown in the block diagram in Figure 4.2.

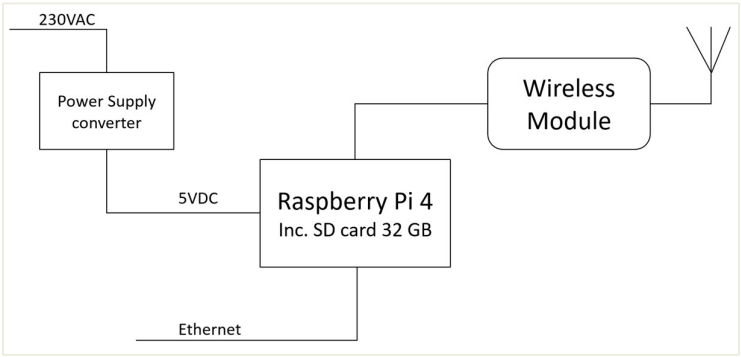


Figure 4.2: Block diagram showing the hardware connection diagram for the application PM.

A proof of concept was conducted and demonstrated utilizing a simulated program created with Node-RED that simulated the air cylinder, VFD, and sensors. Grafana was used to create a dashboard to visualize the simulated data. Literature reviews of data security in cloud-based systems, as well as evaluations of the IEC 62443 and the NIST standards, were conducted. As part of the security measures, several security measures were specified for the application to

implement. ML models were briefly discussed from the standpoint of how they can be utilized for maintenance purposes from within the application.

4.1.2 Further analysis

In this section of the analysis phase, further analysis at the beginning of the development process of the application, including data presentation on the dashboard and the choice of the cloud-based database provider during this Master's thesis, was carried out. In addition, the security aspects of the cloud-based database are also presented.

4.1.2.1 Data visualization

As stated in the prior analysis that data from air cylinders, VFDs, and other sensors are presented in dashboards using Grafana as a visualization tool. The dashboards consist of a graph and stats panel for presenting the trend of data in the graphs and numeric values using stats panels. The value for air cylinders will be shown as a percentage of its live service rather than the actual value, which is the distance of strokes in kilometers. This was a request from the Moss department since presenting the percentage of the live service rather than the value of the air cylinders will be more meaningful to the operators. The scaling, for instance, an air cylinder with live service from 0-10,0000 km will be scaled to be between will 0-100 %. Presenting values for VFDs will still be their actual values which are running hours (Hr). As identified in previous analyses, other sensors refer to any sensor that can be directly connected to the WizX (hardware). As part of further analysis with Moss, it was suggested that a pressure sensor to measure the pressure of the Compressed Air tank that supplies pneumatic air cylinders be incorporated into the packaging machine. The goal of using this sensor is to detect the leakage from the compressor air supplier itself or within the connection between the valve manifold and cylinders. Early detection of leaks will greatly reduce the amount of energy consumed, CO_2 footprint and maintenance costs. Since an actual pressure sensor will not be included in this master thesis, a simple program will be created within the application that simulates a pressure sensor generating values between 0 and 100% for the volume level of the compressed air supplier.

Instead of using row panels as presented in the proposed dashboard in the prior analysis report [2] to create levels on the dashboard with different pages for the packaging machines, three dashboards will be created where one is the main dashboard (Home) that includes the common alerts information for the dashboards and list of to the other two dashboards that are linked to the Home page as it structured in the in Figure 4.3, which shows the Home dashboard with a list of dashboards for Packaging Machine 01 and 02. Below the list, for dashboards, common alerts will be presented with different lists with alert information. Packaging Machine 01 will present The [9]Test Celle machine, while the Packaging Machine 02 presents a simulation packaging machine. The simulation dashboard is similar to the Test Celle dashboard and was created only for the purpose of testing the application during the development since the authors of the master's thises main location is in Porsgrunn while the Test Celle is located in Moss.

[9] Test Celle machine is a prototype for the packaging machine that consists of a robot, two air cylinders, two VFDs, and a PLC. It is located in Moss manufacturing hall and is used to test program logic during packaging machine construction.

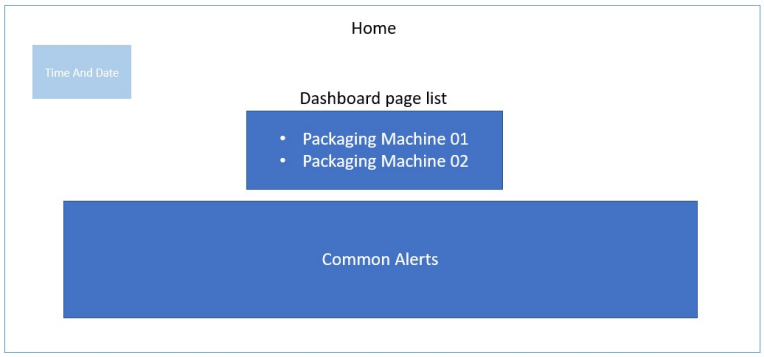


Figure 4.3: Structure of application packaging machine dachboards in Grafana.

Implementing alerts with email notifications

The application requirements document specifies that email notifications and alerts from dashboards are also required [2]. Grafana allows alerts to be created and viewed on the dashboard. Additionally, it allows for an operator to be notified when an alert arises via the notification email channel [46]. In order to send emails, Grafana uses an SMTP server [47]. To accomplish this, an SMTP server must be created and set up within the Grafana server. This is done by configuring the Grafana Docker container with information from the SMTP server provider, including the email address, password, and an email account to authenticate with the email service provider. A free SMTP server provided by the Elastic Email company service provider for application PM is used. This service allows a user to create free for using SMTP servers with limited capacity for sending emails. An illustration of how to create alerts and email notifications using the graph panels and the data flow for displaying alert states and e-mail notifications can be seen in Figure 4.4. Data is retrieved from the PostgreSQL database in real-time by the graph panel.

Alert rules and alert conditions are configured on the graph panel and stored in the Rule Engine. Email notifications are configured with an email account in the Grafana server. The last value from the graph panel of the queried real-time data is compared to the threshold stored in the Rule Engine. If the last value is not above the threshold for 10 seconds, the alert state will be in the 'OK' state (green colored heart symbol). If the last value is above the threshold, the alert state will change its state to the ‘Pending’ alert state (orange-colored heart symbol). Once the value remains above the threshold for 20 seconds, the alerting state will change to 'Alerting' (red-colored ‘broken heart’ symbol). The process will repeat every 20 seconds, which is the time that can be modified to a lower or higher value.

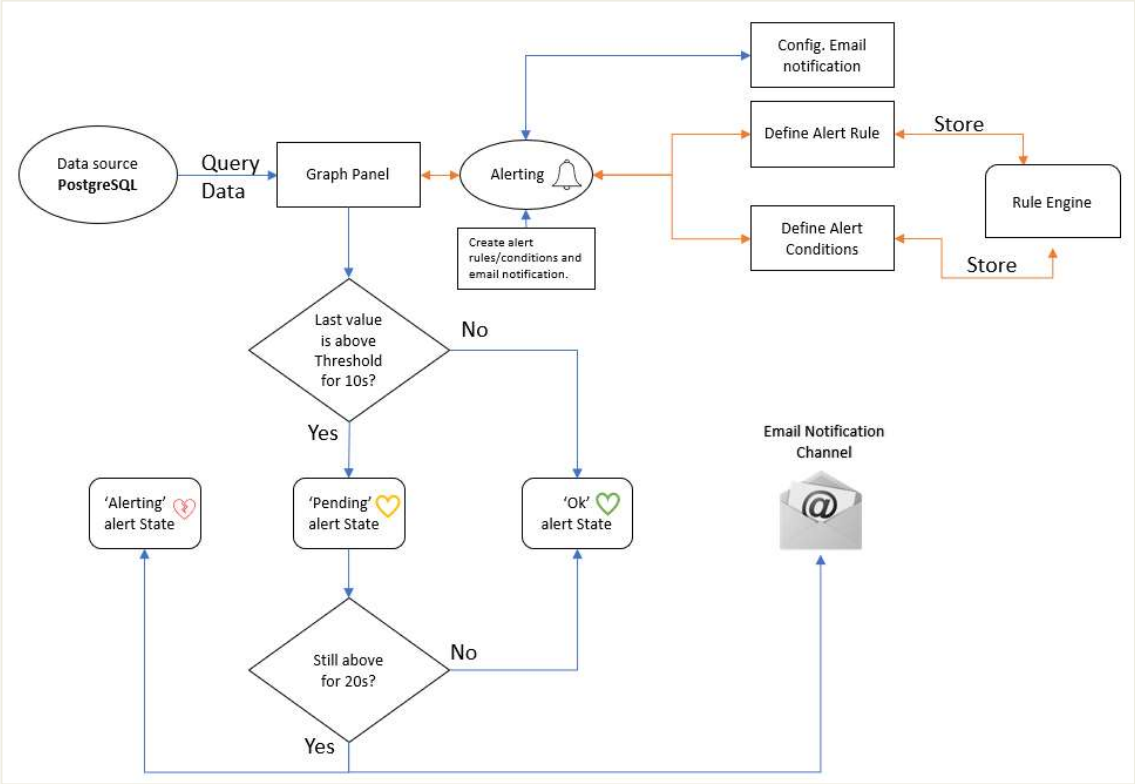


Figure 4.4: Flow diagram shows the configuration of alerts and email notifications for the Application PM.

4.1.2.2 Cloud-based Database Modelling

As defined in the use case diagram from the Figure 4.1 Database, it is one of the actors that interact with both Collect & Store Data and Maintain Configuration uses cases. For readers' clarification, this database presents the cloud-based database. The goal of using a cloud-based database system service is to store historical data from the Application PM that can be used to analyze and develop ML models to predict the maintenance of packaging machines to ensure maximum operating performance.

Currently, there are a variety of cloud computing and hosting providers, such as Microsoft Azure, Amazon Web Services, and Google, that offer VMs, Web Servers, and Database systems to customers on a monthly basis. As a Cloud Computing and Hosting platform, Microsoft Azure is selected for this application. Microsoft Azure offers many database solutions like cloud services, such as relational databases, NoSQL databases, and object stores. These services are fully managed services where the customer doesn't have to do anything with the underlying infrastructure of these databases. They are offered with different features and different pricing models. Thus, during the analysis for choosing the database that fits the requirements for this application, the following features were analyzed:

Security: Encryption of data, which means the data that is stored in the database to be fully encrypted. So that if the disk is stolen, they won't be able to access the data. The other important

aspect of security is network isolation. This means no one can access the database unless they are given access to it.

Backup: What kind of data backup there is, and what retention period. What is the frequency of the backup, and how long will this data be stored in Azure before it is deleted?

Availability: Service Level Agreement (SLA), Replication, and Disaster Recovery (DR).

Within Azure, there is also the option to install a database on VM [48]. The pros of using a database VM as opposed to using managed service are full flexibility meaning that a customer can install any edition of the database and configure it in the way that customers want. Also, customers have full control not only over the VM but also over the database itself. When it comes to the cons of the database on VM are that customers have to take care of everything such as the SLA, security, availability, backups, and updates to VM and the database. However, a fully managed Azure Database for MySQL Server deployment is chosen for this application.

Microsoft Azure for MySQL Server

The database system for MySQL is fully managed Database-as-a-Service by Microsoft Azure and provides database service based on the Community Edition of MySQL.

To use any of Azure services, one needs to have a Mircosoft account to log in and a subscription. For Application PM, the free student account (but limited) provided by the University of South-Eastern Norway cloud not be used because of its limitation in the subscription. Instead, a private account and subscription were used. On the Azure platform, there is more than one deployment option available for MySQL; Platform as a Service (PaaS) and Infrastructure as a service (IaaS) options. PaaS is a provision of a software platform (development & deployment) as a cloud computing over the internet [2]. As a deployment service, PaaS has two deployment options; Azure Database for MySQL – Singel Server and Azure Database for MySQL – Flexible Server. The Single Server deployment option provides a fully managed database service which means it handles most of the database management tasks, i.e., patching, backups, high availability, and security that is automatically handled by database management service. This deployment option provides 99.99% availability on a single availability zone [49]. Compared to the single Server, the Flexible Server option gives more control over database management tasks and better cost optimization controls (i.e., start/stop server) [50]. When it comes to the IaaS option for MySQL, there is only one option: using MySQL on Azure VMs. Comparing the MySQL Deployment options notes that there are significant differences between the PaaS and the Iaas option. For instance, in the PaaS deployment options, many tasks are managed automatically maintained (i.e., backups, MySQL, and OS patching), whereas, in the IaaS, all these are maintained by the user. Also, some scaling features are not supported in the IaaS deployment. However, for this application, the deployment option MySQL – Single Server as PaaS is selected cause of the simple database design for the application, and it is cheaper.

MySQL

MySQL is an open-source Database Management System (DBMS) under the GPL license Relational Database Management System (RDBMS). It supports the Structured Query Language (ANI-SQL) and provides extensions to (ANSI-SQL). It also has various editions such as MySQL Embedded (OEM/ISV), MySQL Classic Edition, and MySQL Community

Edition). MySQL Community Edition is used for this application because it is a freely downloadable version of MySQL and supports SQL and NoSQL databases [51]. For creating and managing the database (scheme) for this application, MySQL Workbench is used. It is an open-source client tool and includes the Community Edition of MySQL.

The Application PM database may vary from customer to customer in terms of the number of machines and types of machines, but the structure should be the same. Each customer will have one database, which can contain one to several machines. The machine contains many cylinders, VFDs, and other sensors that are specific to the machine. The model of a customer database is illustrated in Figure 4.5 by a block diagram giving an overview of the information included in it.

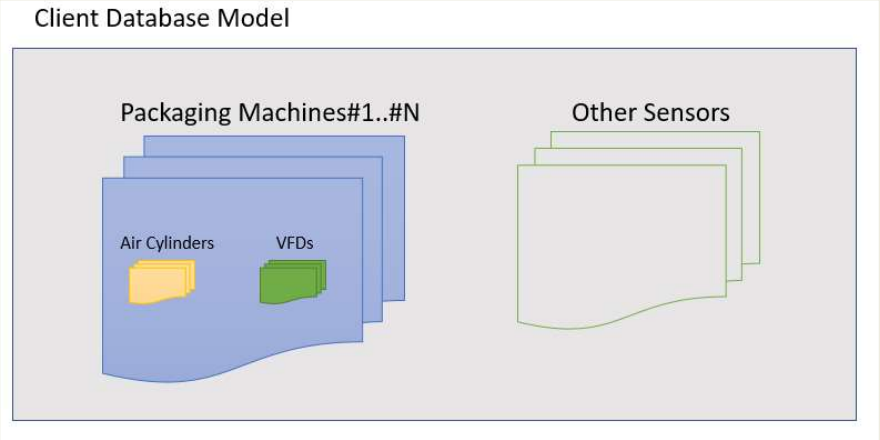


Figure 4.5: An overview of a client database model

Database Scripts

For creating the cloud-based database, different database scripts have been used. This makes it easy to install a new database or update the existing database where one just executes the script or does some changes and then executes it. Another benefit is properly documenting the database documents, making it easy for other developers to have the latest database script and run it. Modeling features in MySQL are used for designing the database and creating tables. The database with tables and an SQL script is generated from the designing and modeling. The scripts are run in a specific order to work. The script for creating databases and tables is in the first row, and then the Default Data Scripts should come in the second, and different other SQL query scripts for analysis purposes as they can be found in Appendix C.

4.1.3 Security in Microsoft Azure Database

Security is one of the most important parts of every system. As a cloud-based services provider, Azure offers a lot of security measures for its resources [52]. As a client, it is important to use those measures and follow security patterns to avoid security incidents. Thus, this subchapter presents the analysis and implementation of the security measures steps that are taken for the application database server in Azure.

Under the security management category in Azure, many critical features are offered. The following security features are analyzed and implemented in the application Azure Database for MySQL:

- Once the MySQL Single Server database on Azure is created, the next step is to set proper firewall rules for MySQL clients (MySQL Workbench and Application (Node-RED UI)) that will have access to the database, as illustrated in Figure 4.12. This is done by configuring the dynamic IPs for MySQL clients to set up the firewall rule in Azure.
- Enabling Secure Sockets Layer (SSL) connections between the database server and clients. This approach secures the connection and prevents "man in the middle" attacks by encrypting the data stream between the database server and the client [53].
- Enforcing the Transport layer Security (TLS) version 1.2 encryption for connections between the database server and clients.
- Disallow access to other Azure services to the application's database server

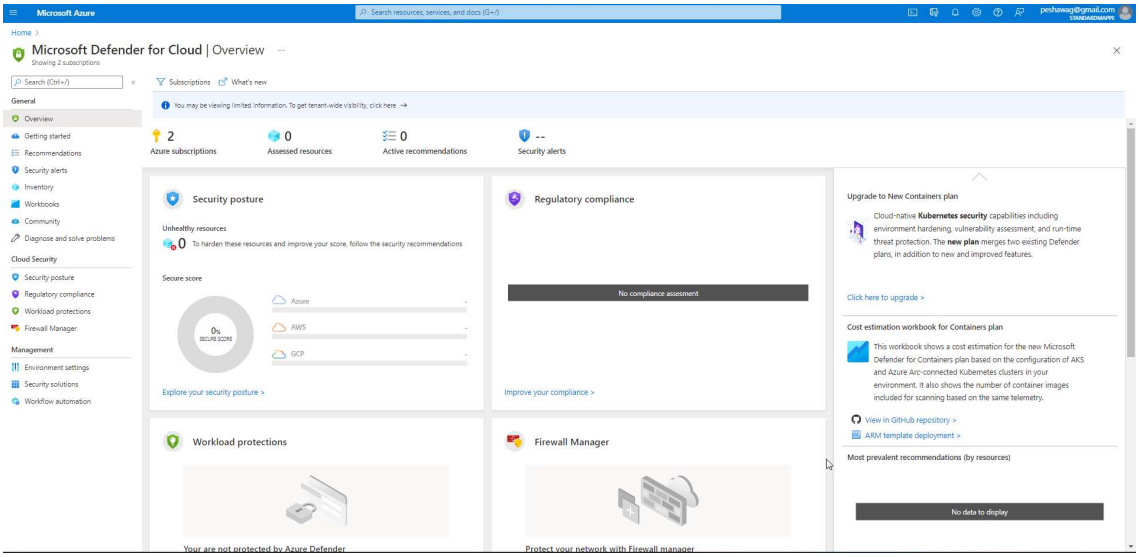


Figure 4.6: Overviuw page of Microsoft Defender for Cloud.

Another feature/service that Microsoft Azure provides is the Microsoft Defender for the cloud [54]. It is a central location for monitoring and alerting security-related issues. It displays a summarized list of problems found in the subscription resources, which in some cases allow a single-click fix. So it is good to take a look at Microsoft Defender once in a while to see if any security issues need attention. Figure 4.6 shows the overview page of Microsoft Defender for Cloud for the application's subscription account.

4.1.4 Disaster Recovery (DR) in Azure

DR is a plan to recover from a complete shutdown of a region due to a disaster such as an earthquake, flood, or war.

In order to set up the DR for the data that are stored in Azure MySQL Server, one has to select a DR site, meaning a secondary region that will function as our primary in case of disaster [55] [56]. The selected secondary region needs to be configured to be ready for activation when necessary. In the case of this application is set up and running in the North Europe region, which will be the primary region. For the DR plan, West Europe can be selected, which will have the same database architecture that was created for the North Europe site. This means that

the primary region is actually used but not the secondary region. In case of a disaster occurs in North Europe, West Europe will be the primary region.

4.1.5 Database Backup and Restore in Azure

Microsoft Azure platform offers backup and restores functionality for MySQL Server. The backups of the data files and transactions are being taken automatically. Backup retention and redundancy options are set in the Pricing tier dialog [56]. During the creation of the database server for this application, the pricing tier dialog appears where one could also define the retention policy for backups and the redundancy option. Figure 4.7 shows the snapshot of a part of the Pricing tier dialogue page where for this application, the Basic Pricing tier is selected where only the Locally Redundant is available and the Backup Retention Period is between 7 to 35 days, where for this application set be 20 days. If the General Purpose or Memory Optimize were selected, the Gengeraficelly Redundant backup redundancy option (as illustrated with text and blue arrow of the figure) would be available for the user to select. This means that the availability of certain features depends on the Pricing tier a user selects. Since this is only for demonstrating the application PM capability of storing data in the cloud database, the basic Pricing tier is selected. When using this application to store data from a customer's packaging machine, the General Purpose or Memory Optimized should be selected.

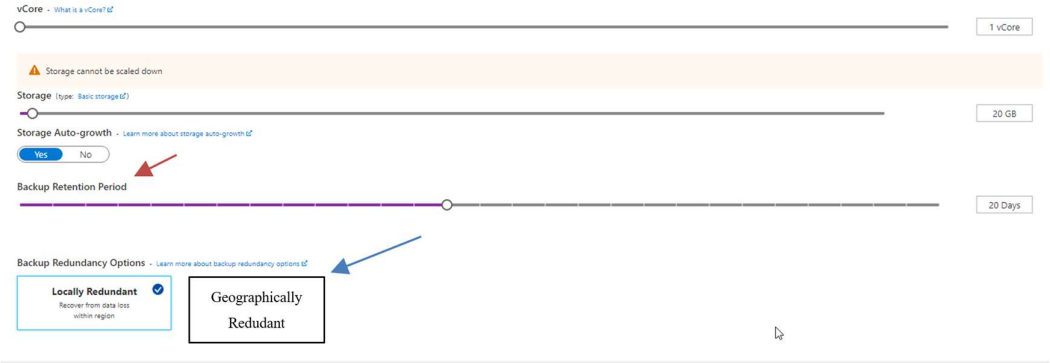


Figure 4.7: Snapshot of Backup Retention Period on the Pricing tier dialogue for basic option.

The Azure platform allows a client to perform a point-in-time restore of the entire server. This means that one must create another server in the same region as the main server if Basic Pricing tiers are selected. Figure 4.8 shows the snapshot of the Restore dialogue where a restore of the stored data can be set from a given restore point (date and time) with the name of the new server, and the location for the new server will be the same as the main server as mentioned earlier. Again, in selecting the General Purpose or Memory Optimized, the restore dialog would have given the option to create the restore server in a different location (region) than the main server. For this application, a restore server is not necessary; thus, it is not created.

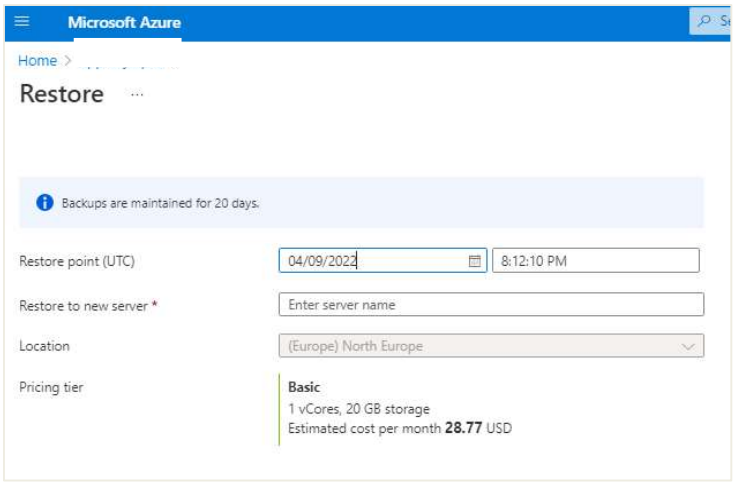


Figure 4.8: Snapshot of Restore dialogue in Microsoft Azure.

4.1.6 Application Components

This subchapter presents the application components to summarize the analysis phase. Figure 4.9 shows the updated application PM components diagram presented in the prior analysis report [2] and the diagram of the components for WizX Core in Figure 2.16. The components are the same as it is defined in the WizX Core in Figure 2.16, but with an extension on the Node-RED Container, TimescaleDB Container, and Grafana Container. Based on the further analysis in this master's thesis, the Node-RED reads air cylinder data in km while the VFDs are hours (Hr) from the PLC, and the air pressure sensor simulator within In Data flow generates random data. The (admin) wizxcore flow scans the updated data from the OPC UA server and updates the local database (TimescaleDB). Also, Instead of Out Data flow, a new flow is created called Azure DB. The function of this flow is to retrieve data from the In Data flow and store them in the Microsoft Azure Database for MySQL Server. The cloud-based database can be configured through the Node-RED UI that is created using the Node-RED dashboard nodes in the Azure DB flow. The collected data in a cloud-based database will be used as historical data for data analysis and developing ML models.

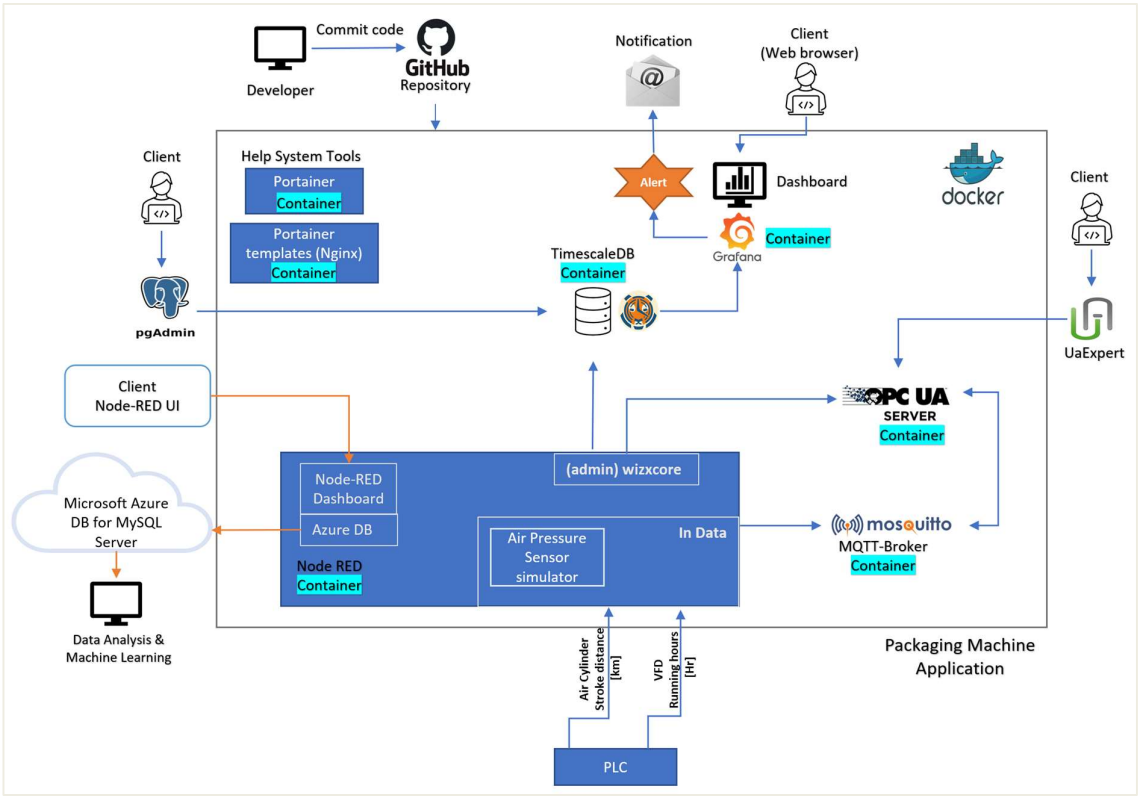


Figure 4.9: updated Application Packaging Machine components diagram [2].

4.2 Design

In the subchapter, the design of the application PM is presented, including the application's data flow and interaction diagrams for each use case using UML diagrams. The design of the local and cloud-based database and dashboard is also presented.

4.2.1 Application PM Structure and Data Flow

Following the analysis phase, the application PM is an extension of the WizX Core Docker-based components with Node-RED runs on the Raspberry Pi 4 and collects data from the PLC. The Raspberry Pi and the Siemens S7 1500 PLC from the Test Celle are connected using the Ethernet TCP/IP protocol. Since a physical pressure sensor is not available for this thesis, a simulated pressure sensor is created within the Node-RED In Data flow which generates random data. The data from the PLC and simulated pressure sensor are prepared and formatted into an MQTT Sparkplug B object and collected into a single MQTT Sparkplug B device/node definition using a sub-flow referred to as PM01 (Packaging Machine 01) becomes one MQTT Sparkplug B device/node containing two devices as illustrated in Figure 4.10. Metrics (tag name, value, and data type) and timestamps are sent from this node to the MQTT Broker (Mosquitto). The collected data are then subscribed by the OPC UA Server that is integrated with the Sparkplug MQTT interface. From the (admin) wizxcore flow, the OPC UA Client scans the OPC UA Server using the browse function for device metrics automatically or manually once a new device has added or changed its data type or tag name. The wizxCore Config updater node gets an array of inputs from the OPC UA browser and updates the taghistory_te table within TimescaleDB. The OPC UA Client 1 retrieves all metrics from the database through the WizxCore subscriber node and subscribes to the OPC UA Server to receive only historical metrics, which are defined to be displayed in the visualization tool (Grafana). The OPC Client then inserts the data into taghistory_data by using the WizxCore-Inserter. Grafana, as the visualization tool, is, then queries the metrics and visualizes them in a dashboard.

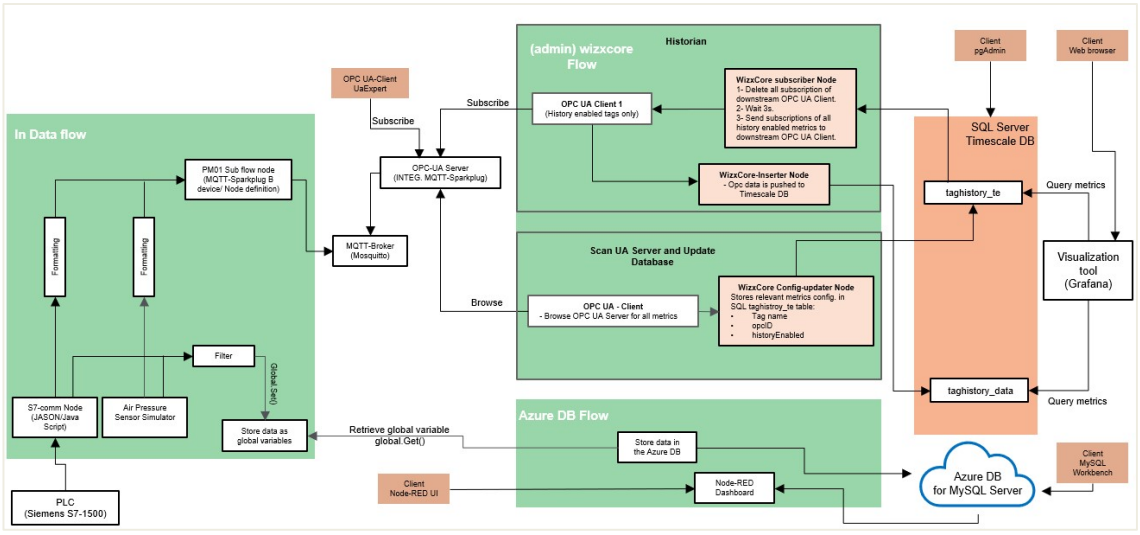


Figure 4.10: Application Packaging Machine structure and data flow for Test Celle machine.

At the same time, that data is read in the In data flow from PLC and the air pressure sensor; those data are filtered and stored as global variables with tag names. The filtering process takes place through the Filter node, which allows data to pass through only when it has changed. In this way, only updated values are stored as global variables, which reduces data traffic significantly when the data are stored in the cloud. The stored global variables are retrieved and stored in the Azure Database for MySQL Server. Data in the cloud database can be monitored using MySQL Workbench as a client and configured through the Node-RED UI. In the same way, the local database can be monitored and configured through the pgAdmin 4 software as a client. There is also a possibility to monitor live data in the OPC UA Server using UaExpert as the OPC UA Client.

4.2.2 Interaction Diagrams

The design phase is based on the knowledge from the use case diagram developed under the analysis phase for designing the application to fulfill the requirements. By creating an interaction diagram for each use case, that represents the dynamic behavior within the application. The interaction diagram consists of the Sequence Diagram (SD) and Collaboration diagram. The Sequence Diagram is a time-dependent dynamic system where it shows the sequence of the messages in time, while the Collaboration Diagram shows the collaboration with a focus on data. Since the application is time-dependent; hence the focus will be on a Sequence Diagram (SD).

The goal of the design phase is to make a design model based on the objects that handle both the system's dynamic and static behavior. These objects have some responsibility to fulfill the goal of each use case that is defined for this application. But since applications are programmed on Node-RED consisting of nodes, there are no objects or classes. Still, an SD is created for each use case that can be found with a brief description in Appendix D.

4.2.3 Local Database Design

TimescaleDB is used to store the time-series data locally on the SanDisk MicroSD 32GB card on the Raspberry Pi 4 attached inside the WizX enclosure. The ER diagram for the database model and the table descriptions are presented in Figure 2.12 under subchapter 2.3.1. TimescaleDB as data source used where Grafana queries metrics associated with a timestamp and visualized in dashboards. Data from the local databases are retained for one week before new data overwrite.

4.2.4 Cloud-based Database Design

A cloud-based database is required to store information from the packaging machines, including air cylinders, VFDs, and simulated air pressure sensors. Microsoft Azure Database for MySQL Server is selected for storing data in the cloud-based database. The MySQL Workbench database tool is used to create a script for tables, databases, and an ER diagram, as illustrated in Figure 4.11. The database model consists of five tables containing information about the users, packaging machines, air cylinders, VFDs, sensors, and log data (logdata) of the packaging machine. The ER diagram shows the relationships between the tables where a non-identifying relationship is used between air cylinders (cylinders), sensors, VFDs (vfds), and packaging machines (packaging_machine) tables. This means that one packaging machine

can have one-to-many air cylinders, VFDs, and sensors. In the same way, a non-identifying relationship is used between cylinders, sensors, VFDs tables, and measurement data table (logdata), meaning that the log data table can contain one-to-many values from air cylinders, VFDs, and sensors table. The user database (userdatabase) table stores the hashed password and username created by Node-RED dashboard/Azure DB flow. The table is not connected to any other tables as it is only used to restrict the Node-RED UI screen, which means that the user registered in the table has different rights to the screen of the Node-RED UI for configuration of the cloud-based database. The script for creating databases and tables is included in Appendix C.

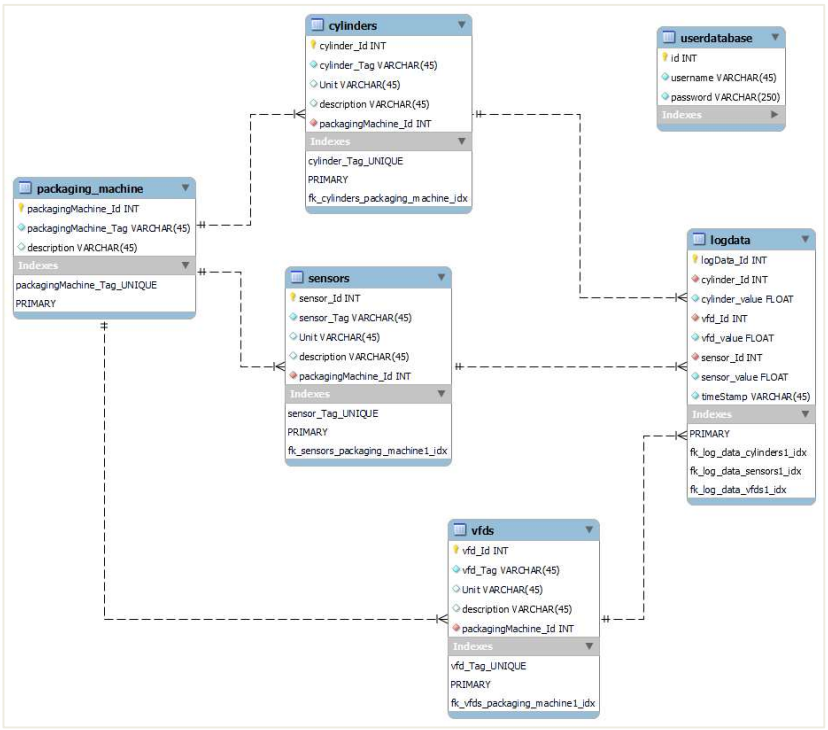


Figure 4.11: Cloud-based Database Model ER Diagram

Cloud-based Database Configuration

As part of the master's thesis task description can be found in Appendix A. The database model for storing data in the cloud should be configured and flexible for adding new content and updating or deleting the tables for packaging machines, air cylinders, VFDs, and sensors. The configuration is done using the MySQL Workbench and via the Node-RED UI, as illustrated in Figure 4.12. The MySQL Workbench has a user admin who is allowed to configurations listed in the green box in the figure. Through the Node-RED UI as a client with access to the MySQL Server database, authenticate users are able to configure the database tables in the cloud. An admin user is created and allowed to all the configurations listen in the green box illustrated in the figure. In addition, there is an operator user that is created by the admin user and authorized to do configuration on the database but not create, update or delete other users. The user's authentications include a username and encrypted password that is stored in the database.

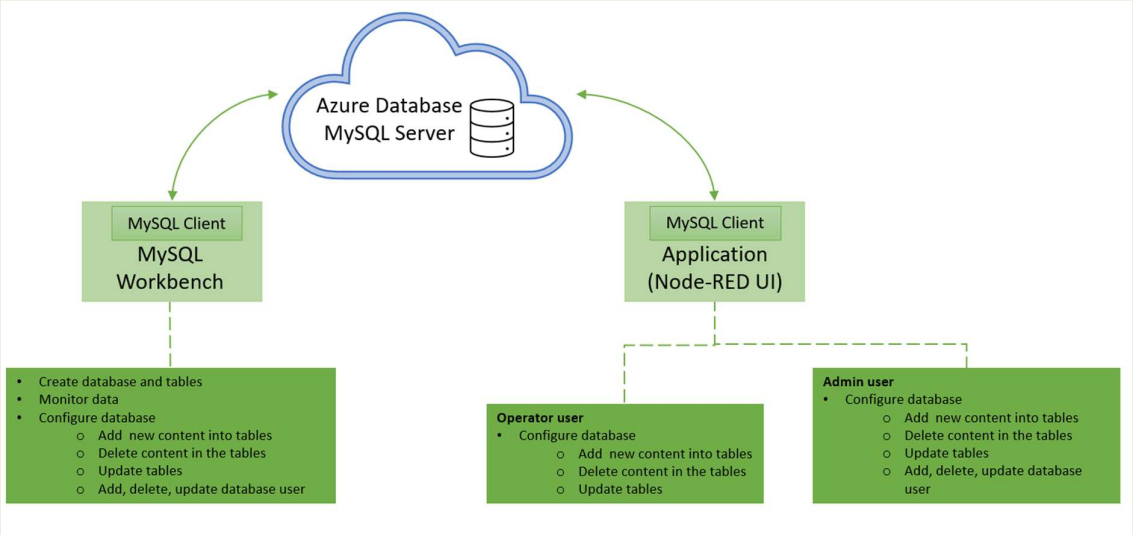


Figure 4.12: System sketch for stoing data and manging in the Microsoft Azure Database For MySQL Server.

4.2.5 Data Visualization

Following the analysis phase, Grafana is used as a visualization tool to create and design dashboards that provide real-time data and alerts collected from air cylinders, VFDs, and pressure sensors operating on packaging machines. With reference to the proposed dashboard structure shown in Figure 4.3 conducted in the further analysis phase, three dashboards are created using Grafana; Home, Packaging Machine 01, and Packaging Machine 02 dashboards. The screenshots of the dashboards are presented in this subchapter, along with brief descriptions included in the report and Appendix E.

Home dashboard

The Home dashboard's aim is present the common alert information from the Packaging Machine 01 and Packaging Machine 02 dashboards. A screenshot with further description can be found in Appendix E.

Packaging Machine 01dashbord

The Packaging Machine 01 dashboard displays real-time data from the Test Celle machine and a simulated air pressure sensor. The Test Celle is a prototype for a packaging machine for program logic testing that is necessary during the construction of packaging machines at the Moss manufacturing facility. It consists of a robot arm, two pneumatic air cylinders, VFDs, PLC SIMATIC S7-1500, and Human Machine Interface (HMI). A screenshot of the dashboard, along with a description and photo of the Test Celle, can be found in Appendix E.

Packaging Machine 02 dashboard

The Packaging Machine 02 dashboard was also created to visualize the simulation data from two air cylinders (A506, A508), two VFDs (A3 and A4), and a compressed air pressure sensor (P02), as shown in Figure 4.13.

The dashboard is designed in the same manner as Packaging Machine 01, in which graphs, stat, and gauges are used to display data in real-time. The top left of the dashboard is the Grafana menu, from which the dashboard and alerts can be configured. The dashboard name with the description of the packaging machine is called the Simulation Packaging Machine in presented using a text panel. The link to the Home and Packaging Machine 01 is listed in a list panel at the top of the dashboard. On the top right of the dashboard, there is a dropdown box that allows the user to pick the time range for which the dashboard displays data. Currently, the time range of the real-time data is cased set to last 30 minutes. The auto screen refresh drop-down time can be selected next to it. In the top right corner of the dashboard, the time and date are displayed on a green background. The rest of the dashboards are derived from three-row panels, one for air cylinders, one for VFDs, and one for other sensors.

These rows can be collapsed to make the dashboards more compact. It is a useful feature during the analysis of one specific row when there are hundreds of panels for cylinders and VFDs grouped together since a user can collapse the other rows and keep the one open for analysis. The cylinder row consists of two time-series graph panels, two stat panels, and one graph panel that displays real-time data from both cylinders.

Time series panels display the name of the cylinder (tag name) and the description of the panel. Similarly, VFD and pressure sensors panels display the same information.

Figure 4.13: Screenshot of Packaging Machine 02 dashboard with yellow-colored annotations pointing to various parts of the dashboard.

Panels for cylinders show a graphical representation of the cylinder's current operation life service in percentage. The stats panel displays numerical values, whereas the time-series panel shows trend data based upon the real-time values. The x-axis is for a time in (hh. mm) and the y-axis in percentage (%). The color of the background on the stat panels and trend lines on the time-series panel corresponds with the alert state of the device. Once the alert state is 'Ok,' the color of the stat panel and the trend line on the time-series panel is green. When the devise value enters the 'Pending' alert state, the background and line color will also change to orange, while when the device is in the 'Alerting' state, the color will turn red.

In addition, as indicated by the yellow text and arrows, the heart symbols and dashed lines show a change in alert state as indicated by their changing colors. In the same manner, the time-series panel for the pressure sensor is configured. When it comes to the gauge panel, alert states are presented with both numeric and color-coding. From 0 to 90 %, the color green appears, followed by orange from 90 to 95 %, and red between 96 to 100 %, where green indicates the alert state of 'OK,' orange indicates the alert state of 'Pending,' and red indicates the alert state of 'Alerting.' Appendix E provides additional detail regarding the configuration of alerts and notifications within dashboards.

4.2.6 Security in the application

The security in the application encompasses the network, application, user, and database modules. In that regard, to secure the application, all those areas must be secured at

Securing log in and remote access to the application

Following the prior analysis for securing the PM Application login carried out in the preliminary project [2], different steps were analyzed and suggested to be implemented for better security of the application. UFW Firewall has been installed and configured on the Raspberry Pi 4 (WizX-hardware). Furthermore, the Google Authenticator is integrated to implement Two-factor authentication for SSH access and Docker Hub login. The port number for remote access is changed to ensure that access to the app is more secure.

Local database (PostgreSQL Security)

Database administration, authentication, and authorization

The local database is used to monitor real-time data using pgAdmin 4, as described in the analysis phase. Thus, having administrative access to the database would ensure the security of the information within it. This is accomplished by implementing a database design with structured security levels for different user accesses through pgAdmin 4 and Grafana query editor. The administrator, who has full access to the database, has the ability to grant or revoke access privileges for users. PostgreSQL resolves this issue by providing multiple levels of access. In PostgreSQL, six levels of security access are defined, as illustrated in Table 4-1 [57]. The Instance level is the highest privilege security level. A second level is the database level, which entails everything related to the database, but cannot manage users or roles. The third level is the Schema level with control controls; everything comes beneath it, tables, columns, and rows. In the table-level security, access privileges can be assigned to the user based on the individual tables with a selected statement such as selecting, inserting, updating, deleting truncate and trigger. In the column and row level of security, a user can access privileges to individual columns and rows.

Table 4-1: PostgreSQL Secuirty Levels

Security Levels	Description
Instance Level	Users, roles, databases creation, login, and replication
Database Level	Connecting, creating schemas, and more
Schema Level	Using schema and creating objects inside a schema
Table Level	Selecting, inserting, updating, and more
Colum Level	Allowing or restricting access to columns
Row Level	Restricting access to tows

The security in PostgreSQL is set by roles which means that privileges and granted to users. For the application PM, in addition to the admin user, which has the Instance Level of security, an operator user is crated with a Table Security level with only the SELECT statement granted. The operator user is specified and configured in the PostgreSQL data source configuration in Grafana.

Database user permissions for the data source in Grafana

The local database on the application is used as a data source for Grafana to query data and display it in dashboards. In Grafana, the data source configuration is done with a specified database user called operator with restricted permissions, i.e., SELECT permission is only granted to queries from taghsitoty_te and taghistoty_data tables, and DROP TABLE and DELETE FROM statements are not permitted. This is to enhance the database security and better control in the Grafana query editor.

Secure user logins and restrict access to Node-RED's UI

As a result of the further analysis phase and the SD for Maintaining Configuration via Node-RED found in Appendix D, a user with access via the Node-RED UI can access the application's cloud database and perform configuration. In order to authenticate users for logging into the application database, a database user table is created within the packaging machine databases, as shown in Figure 4.11. It contains a user id, username, and hashed password column. The passwords are designed to be encrypted using the bcrypt node[58]. The node uses the Bcrypt hash function, which employs a one-way hash of the password and is based on the Blowfish cipher algorithm. Bcrypt generates and stores a random salt with the encrypted password. Therefore, it is evident that different encoded results can be obtained from the same string.

In order to restrict access to the Node-RED interface, a UI control is designed and implemented as part of the NODE-RED dashboard program flow to control the authorization of users. When a user is logged in as an administrator, the Database Settings and User Settings tabs should appear on the user interface. In contrast, the UI will only display the Database Settings if the user is an operator.

User authorization and dashboard administration in Grafana

In order to improve the security, control, and administration of Grafana dashboards, it is possible to create multiple organizations and teams, each with multiple users with varying access levels. In the context of this application, an organization with one team with multiple users and a set of roles can be created for Goodtech employees. In addition, another organization for packaging machine customers can be established in which users have limited access rights to Grafana dashboards. Users in the Grafana organization role are granted access to the following types of roles [59]:

- **Admin**: For managing data sources, teams, and users.
- **Editor**: For creating and editing dashboards.
- **Viewer**: For viewing dashboards.

Grafana users for the application should consist of one organization, one team, and three users; an admin, an operator, and a guest, based on the above roles. The admin user has the admin organization role, the operator user has the editor role, and the guest user should have the viewer role. A user can be assigned to a team with a viewer role for the same dashboards for the customer organization, as shown in Figure 4.14.

Appendix E contains screenshots taken during the authorization and administration roles configuration for users within the Grafana application dashboards.

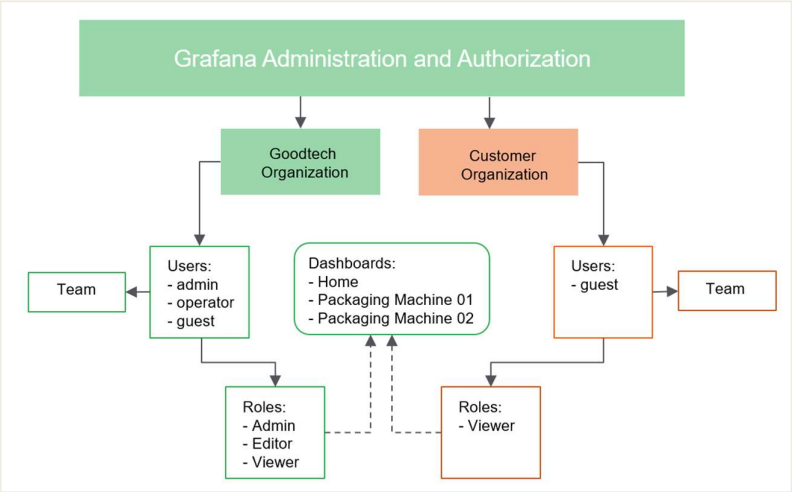


Figure 4.14: Overview of Grafana administration & authorization for the Application PM.

5 Results and Testing

This chapter presents the results of the application PM based on the performed analysis and design. It also presents the testing of the application PM based on the test case document performed on the Test Celle in Moss manufacturing hall.

5.1 Results

This subchapter presents the result of the design of the Application PM, where the data flow throughout the application generated by the Simulation Packaging Machine (PM02) is monitored and stored in the cloud-based database.

5.1.1 Simulation Packaging Machine Node-RED flow

Figure 5.1 illustrates the Node-RED flow for the simulated data of a packaging machine with air cylinders (A504 and A508), two VFDs (A4 and A5), and one air pressure sensor (P02) every 10 seconds. The flow simulates float data type values using the inject node that simulates

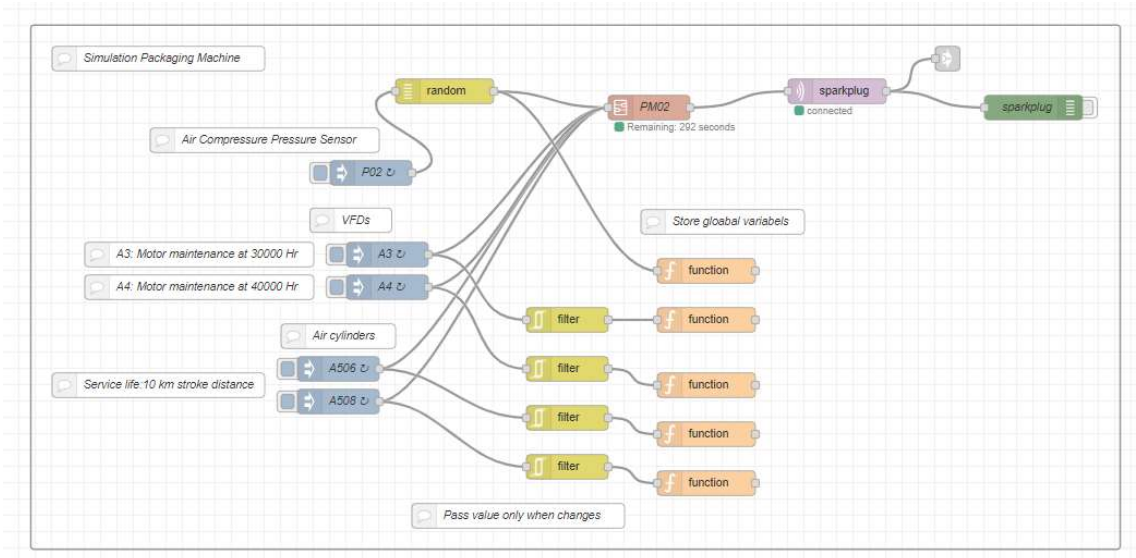


Figure 5.1: Simulation Packaging Machine (PM02) Node-RED flow.

the stroke distance of the air cylinders in km. For VFDs, it generates integer values indicating the running hours in hours (Hr). An inject node generates random float values between 70.5 and 80.1 % that simulate the central compressed air pressure level of the packaging machines. A3 generates data (running hours) up to 30000 Hr, meaning after when it gets to 28000 Hr, the graph (time-series) panel will raise an alert. For the A4, it raises an alert at 38000 Hr. When it comes to A506 and A508, the service life set is 10 km distance of stroke before the air cylinders need to be replaced. Thus, an alert will be at a 9.8 km distance of strokes. The alerts for the pressure sensor are configured to have alert conditions where an alert will arise if the pressure is below 70 or if pressure exceeds 85. The generated data are then connected to the PM02 sub-flow where the data are formatted and published to the MQTT Broker sparkplug B Node.

Results and Testing

To verify the application result with the simulation packaging machine, the distance of stroke for the A506 cylinder set is 8.9 km, as illustrated in Figure 5.2. The testing porpuses the sampling time set to be 10 seconds.

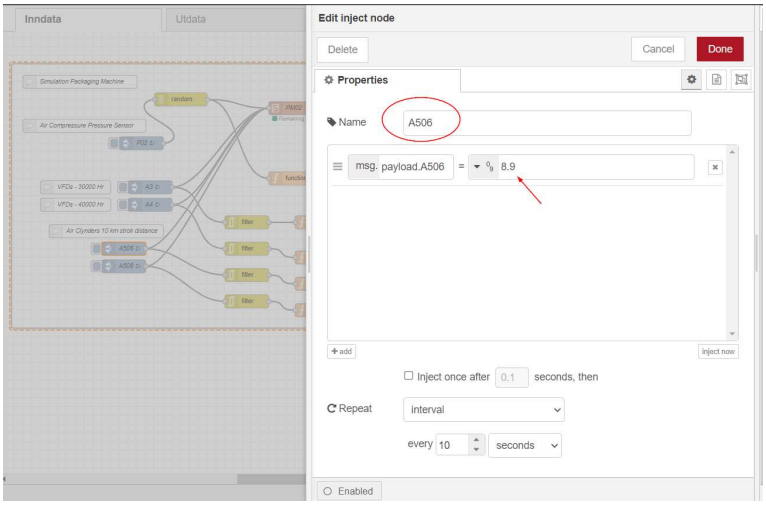


Figure 5.2: A screen shot of the inject node configuration for A506 with an 8.9 km stroke distance and intervals of 10 seconds (sampling time).

5.1.2 Monitoring data flow

The OPC UA server then subscribes to the collected data within the MQTT Broker. From the OPC UA Server, data are then stored within TimescaleDB using the OPC UA Client; for more details on how data are transmitted and stored, see Figure 4.10 or the Sequence Diagram that can be found in Appendix D, as well as the Node-RED, flows in that can be seen in Appendix F. The generated data are then monitored from the OPC UA server using UaExpert. They are also monitored within the TimescaleDB using pgAdmin 4. As an example, the generated data from A506 and A508 are monitored from the UaExpert and pgAdmin 4 by querying the float value column with timestamps and tagid from taghistory_data, as shown in Figure 5.3.

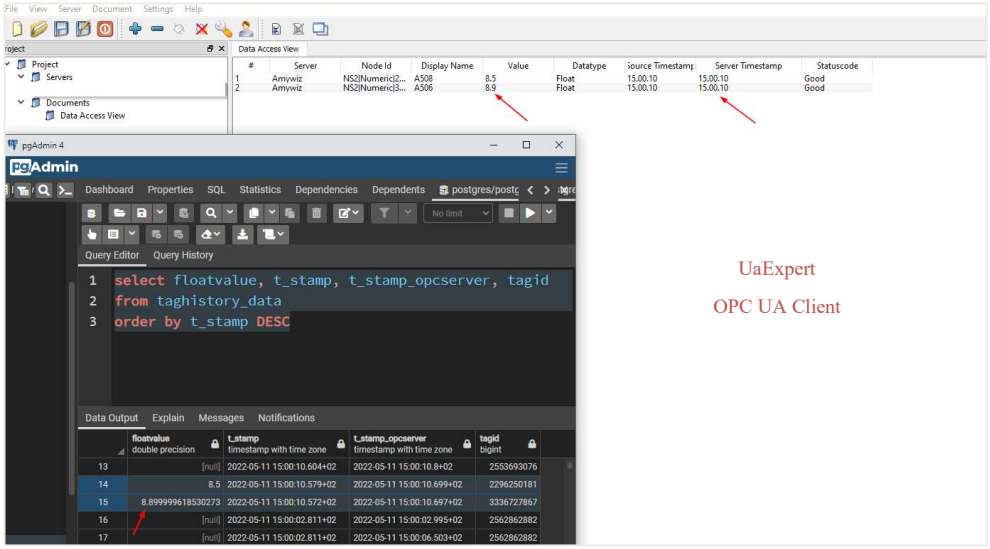


Figure 5.3: UaExpert and pgAdmin 4 screenshot monitoring data from A506 and A508. A506 shows value 8.9 and A508 shows value 8.5 at timestamp 15.00.10.

5.1.3 Presenting data in the dashboard

The stored data from the local database are queried and presented in graph and stat panels, see Figure 5.4. The cylinder distance of strokes values is scaled to show in percentage. As it can be seen in the figure, the value for A506 is shown in the stat panel that 89 % of operated its service life. At the same, it can be observed from the graph panel as well as the last value of the cylinder at the bottom right of the graph.

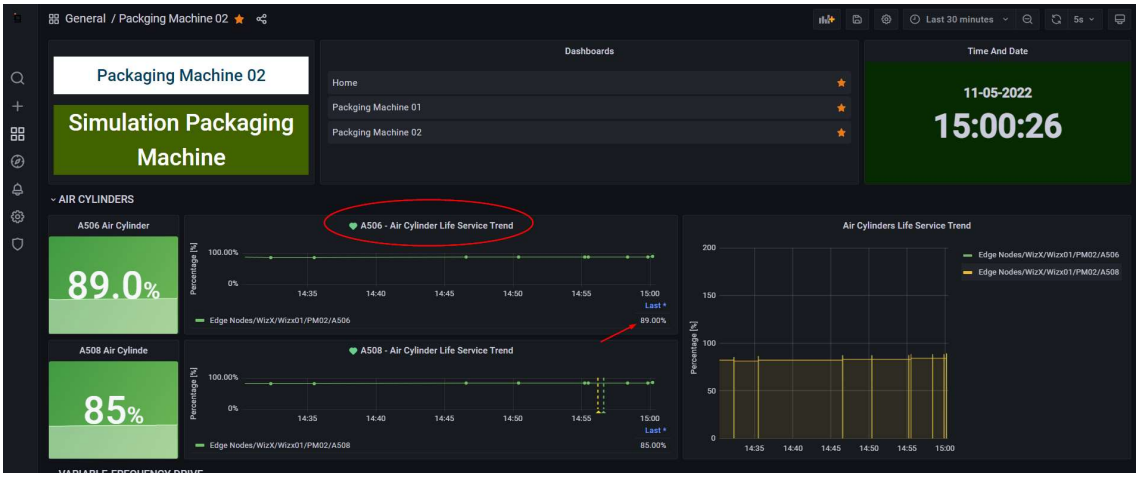


Figure 5.4: Seceenshot of the Simulation Packging Machine dashboard shows the gnerated data for A506 and A508. 89% for A506 and 85% for A508 at timestamp 15:00:26.

5.1.4 Storing data in Azure Database for MySQL Server

At the same time, the generated data is stored as global variables through a filter where only the new value will be stored. The stored data within the global variables are retrieved by Azure DB flow and stored within the Azure Database for MySQL Server. The result of querying the collected data for A506 and A508 with the timestamp from logdata table using the MySQL Workbench can be seen in Figure 5.5.

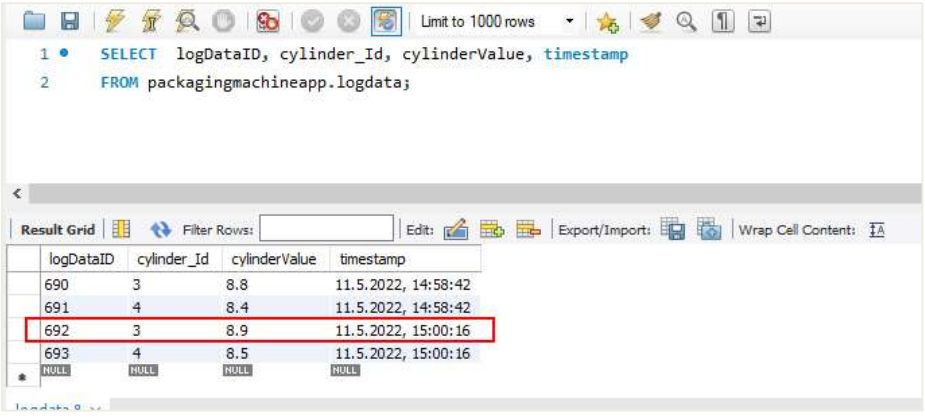


Figure 5.5: Quering stored data for A506 and A508 from the logdata table within the application cloud-based database in Azure.

5.2 Testing

The application was partially tested during its development. However, more systematic testing of the application was performed according to the test case document, which can be found in Appendix G. The test case document includes testing and verification of the functionality of the application and whether it meets the application requirements.

5.2.1 Hardware connection setup under testing the Application PM

During the time of testing the Application PM, a complete packaging machine was not available in the manufacturer's hall in Moss. Instead, the application was tested on the Test Celle machine. A brief description and the actual foto of the Test Celle can be found in Appendix E. Figure 5.6 shows the connection setup between WizX hardware, PLC, and a PC with the application run on it used during the application's testing. Industry Ethernet TCP/IP is used for connecting the WizX hardware to the PLC network switch. A personal computer (PC) was connected to the same network using Google Chrome web browser as clients for Node-RED flow editor, Node-RED UI, and Grafana. The testing was carried out by a Goodtech employee who works at Moss's manufacturing hall.

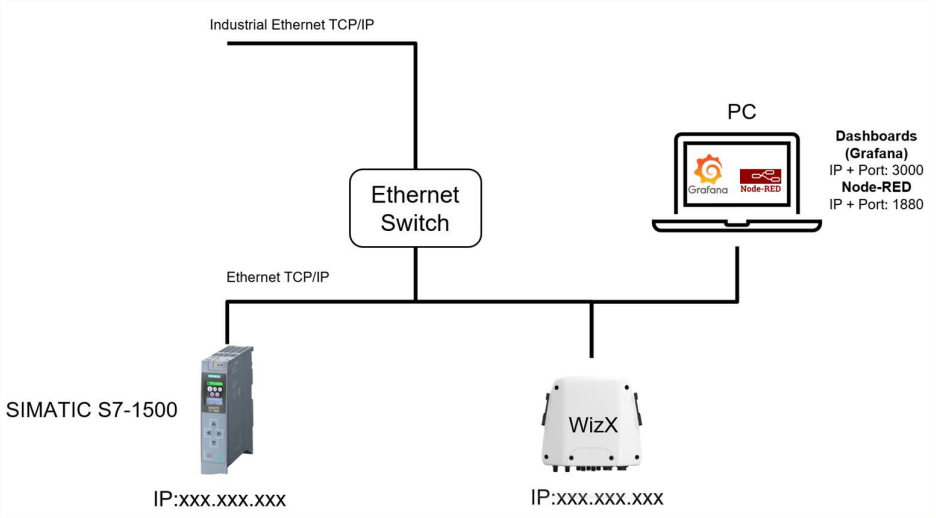


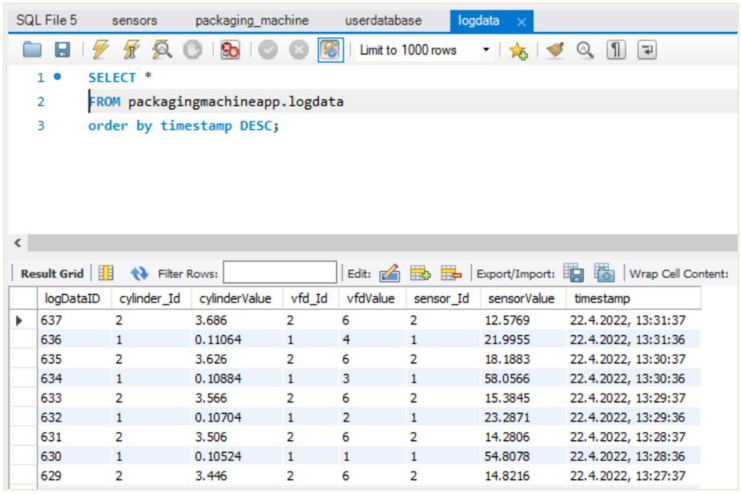
Figure 5.6: Connection setup between WizX hardware, PLC, and PC during the testing of the Application PM.

5.2.2 Collecting and storing data from the Test Celle

Collecting data from the Test Celle machine were tested using the dashboard in Grafana. Figure 5.7 illustrates the screenshot of the Test Celle dashboard, where data from the air cylinders and VFDs, and air compressed sensor simulator are visualized.

Figure 5.7: Screenshot of the Test Celle dashbord. Stat, graph panels for cylinder (A502 and A504) and VFDs (A1 and A2). A gauge panel and graph panel used for compressed air pressure sensor.

The collected data from the Test Cell was stored in the Azure Database for MySQL server every second and was confirmed during testing. This can be seen in Figure 5.8 of the screenshot of the log data queried using MySQL Workbench.



logDataID	cylinder_Id	cylinderValue	vfd_Id	vfdValue	sensor_Id	sensorValue	timestamp
637	2	3.686	2	6	2	12.5769	22.4.2022, 13:31:37
636	1	0.11064	1	4	1	21.9955	22.4.2022, 13:31:36
635	2	3.626	2	6	2	18.1883	22.4.2022, 13:30:37
634	1	0.10884	1	3	1	58.0566	22.4.2022, 13:30:36
633	2	3.566	2	6	2	15.3845	22.4.2022, 13:29:37
632	1	0.10704	1	2	1	23.2871	22.4.2022, 13:29:36
631	2	3.506	2	6	2	14.2806	22.4.2022, 13:28:37
630	1	0.10524	1	1	1	54.8078	22.4.2022, 13:28:36
629	2	3.446	2	6	2	14.8216	22.4.2022, 13:27:37

Figure 5.8: Query logdata table for Tect Celle machine. It shows the collected data from the cylinders, VFDs and compresserd air pressure sensor within timestam between 13:27:37 to 13:31:37.

Results and Testing

In order to verify alerts on the application, the threshold value for air cylinder data (A504) on the Test Celle machine was changed to 20, so an alert is generated, as shown in Figure 5.9. This figure illustrates that an alert has been violated, as indicated by the graph panel's 'broken heart' symbol.

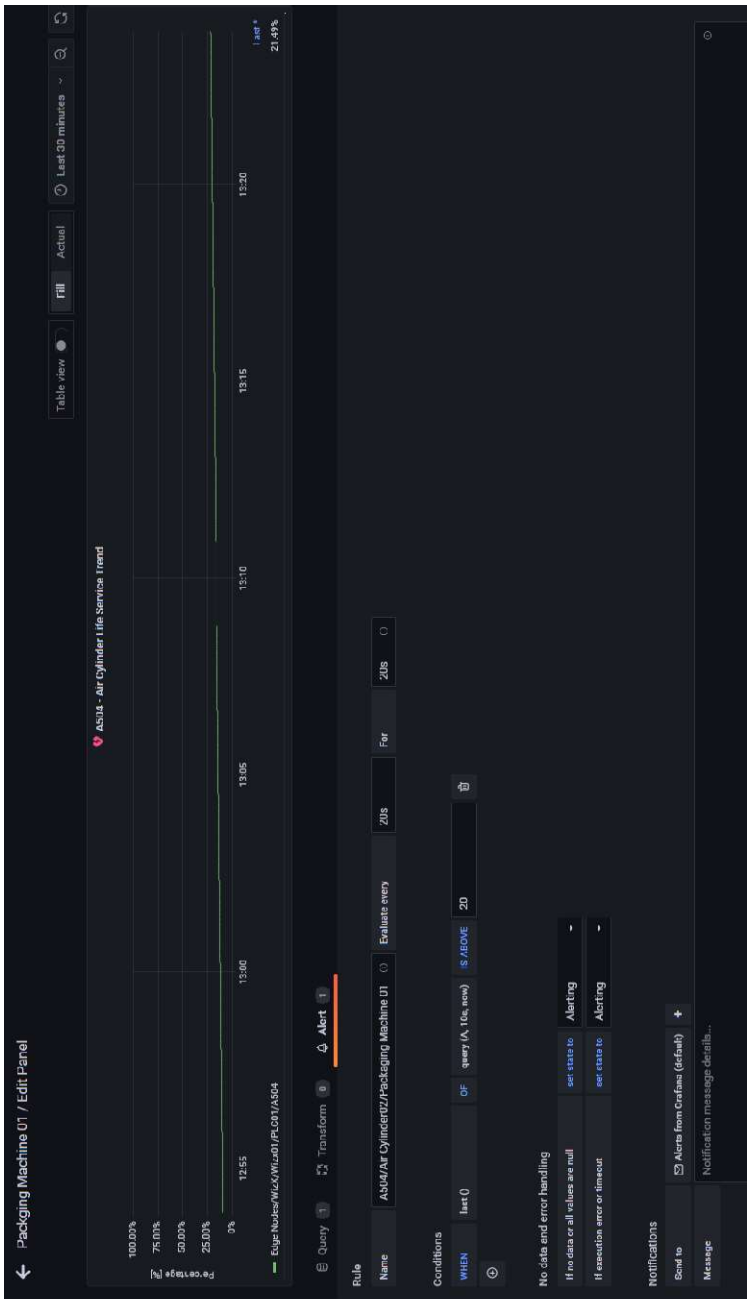


Figure 5.9: Screenshot of the Alert tab of the graph panel for A504. An alert rule is set to be evaluated every 20 seconds for 20 seconds within the condition that the value of the cylinder is above 20 for 10 seconds.

The generated alert from the A504 was also verified and confirmed on the Home dashboard, where all alert information is collected. An illustration of the result is shown in Figure 5.10. The A504 is shown in pending status, and after 20 seconds, the alert state changes to 'Alerting.' It is also possible to view the 'Alerting' state in the Alerting table.

In Appendix H, further testing and results are presented for the application dashboards.

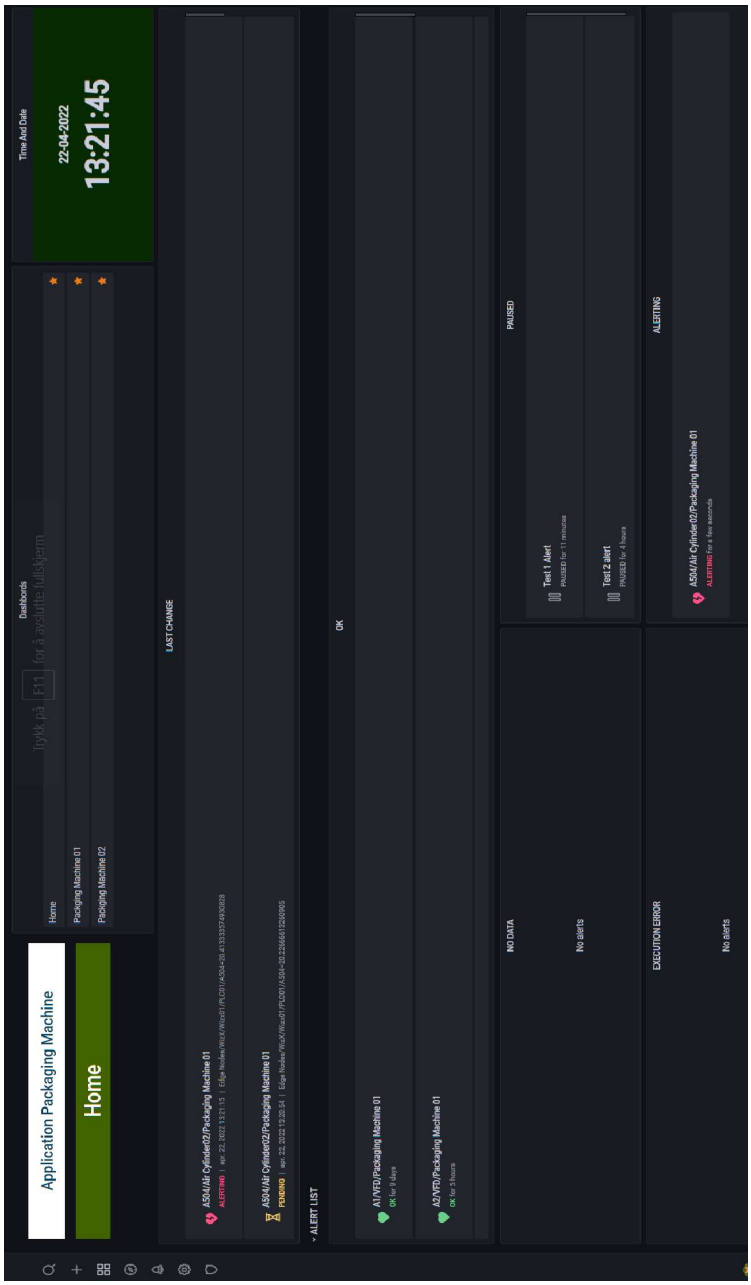


Figure 5.10: Screenshot of Home Dashboard showing both 'Pending' and 'Alerting' states in Last Change Table. The 'Alerting' state is also shown in the Alerting table.

5.2.3 Email Alert notification

The implementation of the process of collecting real-time data from a PLC and a simulation packaging machine and monitoring the real-time data from dashboards in conjunction with alerts and notifications by email whenever air cylinders, VFDs, or air pressure sensors violated the alert conditions were as anticipated. But in the event of an 'Alerting' state, the notification email identified by the email server (Gmail) that was configured as a notification email account was collected by the trash (Søppelpost), as shown in Figure 5.11.

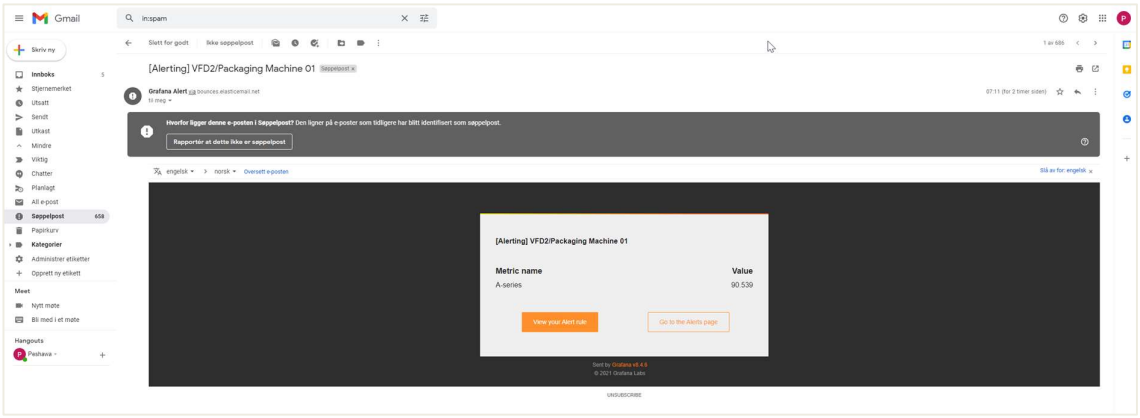


Figure 5.11: A screenshot of the Gmail account receiving an email notification of an alerting state on a VFD found in the trash.

5.2.4 Configuration of Azure Database for MySQL Server using Node-RED UI

The Node-RED UI as a client for connecting to Azure Database for MySQL Server was used during the testing for the configuration of the application database server. Through the Chrome web browser, the Node-RED UI was accessed using the URL with the IP address of the Raspberry Pi/1880/UI. Figure 5.12 illustrates the registration page of the first page, where the stored user name and password must be typed in for login to the database configuration page

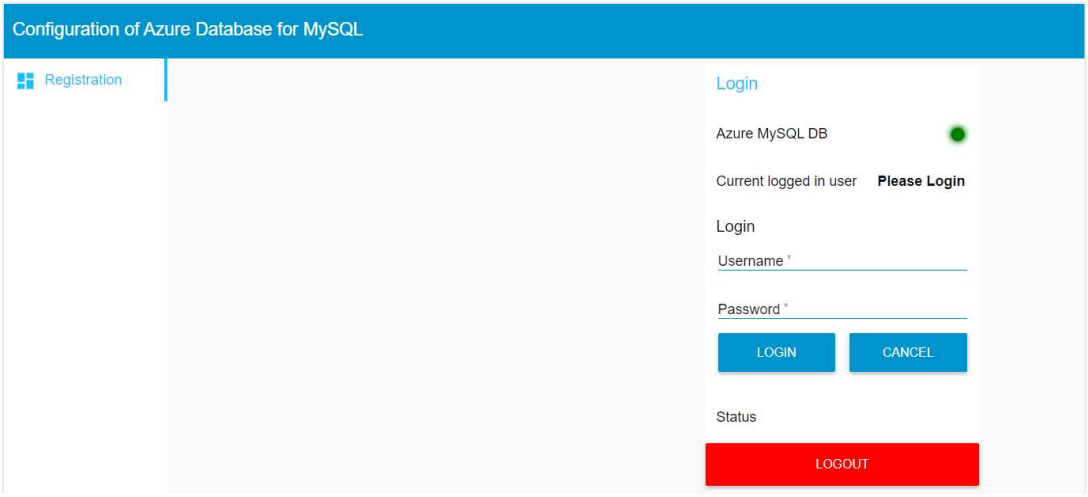


Figure 5.12: Node-RED UI log in page.

window. The green confirms that the database server for the application is connected and running.

In the course of testing, a new simulated packaging machine (PM03) with a VFD (A5) was added, as shown in the screenshot from Node-RED flow and Grafana in Appendix H. The database tables for the packaging machine and the VFD were also updated, as indicated in Figure 5.13. The updating of the tables was checked and confirmed using MySQL Workbench as a client, and the screenshot can also be found in Appendix H.

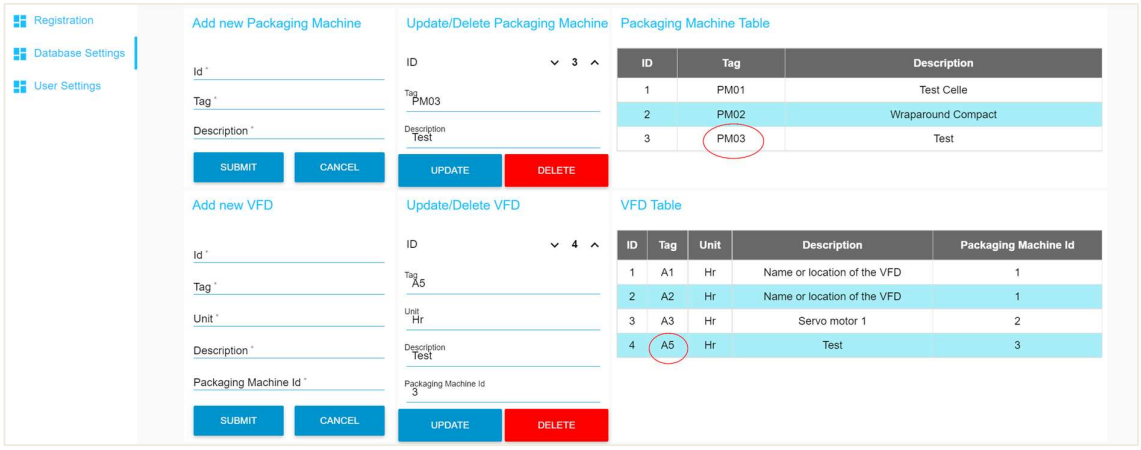


Figure 5.13: Adding PM03 and A5 on the packaging amchine and VFD table within Azure Database for MySQL Server.

6 Discussion

This chapter presents a discussion of the methods used for application and further recommendations regarding being implemented to enhance the security of stored data and the application.

6.1 Securing client access with Azure Database for MySQL Server

Creating a firewall rule on the Azure Database for MySQL server is how to control who can be allowed network access to the application's database server. This is done by configuring and storing the IP address of the client on the connection security setting on the database server. The client's static IP address should be used instead of the dynamic IP address. The reason for using the static IP for the clients instead of dynamic IP is that the dynamic IP address changes, and one needs to update the firewall rule. This will put the database server at risk by allowing the network access to an IP that is no longer in use, which can be used by somebody else. Thus, it is recommended to configure only a static IP address for the clients that connect to the database server on Azure.

For security best practice, it is recommended to use the Virtual Network (VN) and create Private Endpoints connections for Azure Database for MySQL instead of allowing access to the server from the public internet. Setting up the VN requires more configuration and more knowledge of networking. Creating Private Endpoints connections allows connections from within a VN to private IP.

6.2 Configuration of alerts and email notification

The configuration of alerts in graph panels was initially intended to include two alerts per air cylinder. The first one should alert the operator when the distance between the cylinder strokes is close to the threshold. The second one should alert when the value crosses the threshold. After a test of this configuration for 25 cylinders and 25 VFDs, it was noted that the process of alerts became much slower, especially when the evaluation time was measured in seconds. Because every alert rule is stored in the Rule Engine, when the Rule Engine is required to evaluate every real-time data value in seconds, the system becomes slow, and there may be execution errors. Therefore, in the further analysis phase, the requirement to have two alerts was changed to one alert configured for each cylinder, VFD, and other sensors. For the purposes of testing, the evaluation time was set to 20 seconds to receive a quick response. However, the evaluation should be every hour for a real packaging machine because evaluating in seconds is not necessary.

The application sends an email notification when the alerting state appears on the cylinder, VFD, and air pressure sensor. A Gmail account and a free SMTP server were used during the configuration. During the testing of the application, the notification email for alerting was flagged as spam. This occurs because the free SMTP does not have a specifically identifiable

domain name that the Gmail server can recognize and identify as spam. This can be avoided by using a properly configured SMTP server with a recognized domain and port number.

6.3 Pre-processing the historical dataset for ML Models

It is assumed that enough historical dataset is collected in the cloud-based database. Thus, it is recommended that the historical datasets be divided into normal and abnormal datasets. Then, develop one ML model from each dataset to determine which has the highest probability compared with the live data. This will result in smaller datasets for each model. However, the process becomes faster during this process, and the operator is able to determine the cylinder's condition more quickly and clearly.

An additional solution under pre-processing the dataset is to assist or boost the splitting process of the dataset into normal datasets and abnormal datasets where the operator will create a table in the cloud-based database with notes that describe the time and date and which air cylinder failed.

6.4 Data storage in local and cloud-based databases

The local database within the application is used as a data source for Grafana to query metrics. The data will only be stored for a week, and after that, it will be overwritten with new data. This is because the storage capacity of the application hardware is based on a Raspberry Pi, which has a limited capacity of 32 GB. One could argue that the alert functionality of the application could have been implemented as part of designing the local database, but again, in this case, it will put more pressure on the hardware and can cause the application to crash.

For the development of ML models, it is necessary to store data in the cloud. As a result, Goodtech will be able to offer their customers an additional service in which they can optimize the performance of air cylinders within their packaging machines. However, this will also take man-hours, machine learning skills, and a monthly feed for data storage in the cloud that Goodtech has taken into consideration. The bigger challenge is to persuade the customer that production data from the packaging machines will be stored in the cloud. It is therefore important to understand all the security measures Microsoft Azure provides and the security features incorporated in an application when presenting it to a customer.

7 Conclusion

This chapter presents the conclusion and future work improvement.

7.1 Conclusion

During this master's thesis, the continuation of the development of predictive maintenance applications has been demonstrated for packaging machines manufactured by Goodtech As in the Moss Department. The application PM's key aim is to monitor the air cylinder stroke distance, the VFD running hours, and other sensors. This monitoring consists of reading the distance of cylinder stroke from the PLC using the WizX Core and utilizing the supplier's life service data to inform the operator when the distance of the cylinders is close to its operating life expectancy. By reading the running hours of the VFDs on the machines, regular maintenance can be performed on the motors. In addition to monitoring data from other sensors, such as air pressure, measuring the compressed air level in the cylinder would also provide the operator with information concerning the status of the packaging machine.

The prior analysis and a PoC of the Application PM were carried out as a preliminary project in the fall of 2021. An additional analysis was conducted, including storing data in Microsoft Azure for the MySQL server and analyzing the security of the Azure platform. Following the analysis, the application's design was structured by creating SD for each use case describing the system application's dynamic behavior based on time.

For collecting data from the PLC and other air pressure sensors, during the design phase of the application, the extension of WizX Core software, including Node-RED, was used for collecting data from the PLC and air pressure sensor. Grafana was used in order to display and visualize the collected data numerically and graphically in a dashboard. Node-RED was also used to store the collected data in the Microsoft Azure Database for MySQL Server.

The Application PM was tested on the Test Celle located in the manufacturing hall in Moss. The result of the testing of the Application PM was as expected and fulfilled the requirements.

In addition, a study of ML in general and a suggestion for how ML could be applied in predictive maintenance (PdM) application for packaging machines are included.

7.2 Future work

In order for Goodtech AS Moss department employees to utilize this application, in addition to this report, a user manual should also be created for how to use the application.

The Application PM was developed based on the Test Celle, since a complete packaging machine was not available at the time of development. The Test Celle contains only two pneumatic air cylinders and two VFDs. Therefore, in the next version of the application, the

number of cylinders and the number of VFDs should be determined based on a complete packaging machine.

To ensure that the alert email notification works properly, an SMTP server with a recognized domain and port must be implemented. It is recommended to use the Goodtech Outlook SMTP server.

A new Mircosoft account and a subscription for the Application PM must be created to create the MySQL Server databases on the Azure cloud. The database servers within the Azure cloud should be separated, where each customer should have its database named after the customer name.

References

[1] Goodtech AS. "Packaging Machines." https://www.goodtech.no/en-gb/services/industrial-technology/packaging-machines/ (accessed Jun 10, 2021).

[2] P. Galali, "Analysis of a Packaging Machine Application Monitor," Faculty of Technology, Natural sciences and Maritime SciencesCampus Porsgrunn, Project report November 19 2021.

[3] N. L. Pool. "Hva er forskjellen på D-pak og F-pak?" https://nlpool.no/nyheter/d-pak-og-f-pak/ (accessed Mars 1, 2022).

[4] Goodtech AS, "Funksjonsbeskrivelse as WA Maskin," Unpublished, 2019.

[5] M. Hauge. "Goodtech skal levere nytt digitaliseringsprosjekt for Glasopor." https://www.mynewsdesk.com/no/goodtech/pressreleases/goodtech-skal-levere-nytt-digitaliseringsprosjekt-for-glasopor-3103078 (accessed May 13, 2022).

[6] S. Simic. "What is Docker?" https://phoenixnap.com/kb/what-is-docker (accessed Mars 1, 2022).

[7] Raghav. "Docker Beginner Tutorial 2 - How DOCKER works | Docker Architecture." https://www.youtube.com/watch?v=0e-KiGJIiDc&ab_channel=AutomationStepbyStep (accessed Mars 1, 2022).

[8] B. Bashari Rad, H. Bhatti, and M. Ahmadi, "An Introduction to Docker and Analysis of its Performance," *IJCSNS International Journal of Computer Science and Network Security,* vol. 173, p. 8, 03/01 2017.

[9] A.-W. Shihadeh. "Containers: Portainer Review." https://betterprogramming.pub/portainer-review-382575dabb76 (accessed Mars 2, 2022).

[10] P. Kumar. "Node.js for Beginners: How to Get Started." https://www.simplilearn.com/nodejs-for-beginners-article#:~:text=js%3F-,Node.,makes%20it%20fast%20and%20lightweight. (accessed May 12, 2022).

[11] Node-RED. "Packaging a Subflow as a module." https://nodered.org/docs/creating-nodes/subflow-modules (accessed April 5, 2022).

[12] Node-RED. "node-red-dashboard." https://flows.nodered.org/node/node-red-dashboard (accessed April 5, 2022).

[13] M. G. Paul Andlinger. "PostgreSQL is the DBMS of the Year 2018." https://db-engines.com/en/blog_post/79 (accessed Mars 5, 2022).

[14] A. team. "What is pgAdmin?" https://www.adservio.fr/post/what-is-pgadmin (accessed May 14, 2022).

[15] K. Shaw. "The OSI model explained and how to easily remember its 7 layers." https://www.networkworld.com/article/3239677/the-osi-model-explained-and-how-to-easily-remember-its-7-layers.html (accessed May 15, 2022).

[16] C. Bernstein. "MQTT (MQ Telemetry Transport)." https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport (accessed Mars 13, 2022).

[17] S. Cope. "Introduction to MQTT +Sparkplug For IIOT." http://www.steves-internet-guide.com/introduction-to-mqtt-sparkplug-for-iiot/ (accessed April 18, 2022).

[18] B. Kommadi. "Mosquitto : MQTT." https://medium.com/@bhagvankommadi/mosquitto-mqtt-2a352bd8f179 (accessed April 10, 2022).

[19] Node-RED. "A Sparkplug node for Node-RED." https://flows.nodered.org/node/node-red-contrib-sparkplug (accessed April 18, 2022).

[20] S. Hoppe. "The OPC Foundation launches its OPC UAcademics program to supplement OPC UA training in post-secondary institutes." https://opcfoundation.org/news/press-releases/the-opc-foundation-launches-its-opc-uacademics-program-to-supplement-opc-ua-training-in-post-secondary-institutes/ (accessed Mars 15, 2022).

[21] R. T. Automation. "AN INTRODUCTION TO OPC UA." https://www.rtautomation.com/technologies/opcua/ (accessed Mars 10, 2022).

[22] M. K. K. Landsdorf. "node-red-contrib-opcua." https://flows.nodered.org/node/node-red-contrib-opcua (accessed Mars 5, 2022).

[23] J. WALKER. "What is Grafana and When Should You Use It?" https://www.howtogeek.com/devops/what-is-grafana-and-when-should-you-use-it/ (accessed April 10, 2022).

[24] GrafanaLabs. "Plugins." https://grafana.com/docs/grafana/latest/plugins/#plugins (accessed Mars 25, 2022).

[25] H.-P. Halvorsen, *Software Development A Practical Approach!*, https://halvorsen.blog, 2018. [Online]. Available: https://www.halvorsen.blog/documents/programming/software_engineering/resources/Software%20Development.pdf.

[26] Techsparks. "Software Engineering." https://www.techsparks.co.in/software-engineering-as-a-thesis-topic/ (accessed January 24, 2022).

[27] J. Dyson. "Conjoining FURPS and MoSCoW to Analyse and Prioritise Requirements." https://www.linkedin.com/pulse/conjoining-furps-moscow-analyse-prioritise-jonathan-dyson/ (accessed January 25, 2022).

[28] S. Lewis. "waterfall model." https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model (accessed Mars 5, 2022).

[29] K. Rana. "Iterative Model in Software Engineering." https://artoftesting.com/iterative-model (accessed February 15, 2022).

[30] B. Shiklo. "8 Software Development Models: Sliced, Diced and Organized in Charts." https://www.scnsoft.com/blog/software-development-models (accessed February 18, 2022).

[31] R. F. Jr. "Comparing Waterfall and Rational Unified Process." https://theagileblueprint.wordpress.com/2011/03/02/comparing-waterfall-and-rational-unified-process/ (accessed January 30, 2022).

[32] K. Karahainko. "AI vs ML vs DL: differences and use cases." https://mentalstack.com/blog/ai-vs-ml-vs-dl (accessed February 7, 2022).

[33] MathWorks. "Machine Learning in MATLAB." https://se.mathworks.com/help/stats/machine-learning-in-matlab.html (accessed April 13, 2022).

[34] GeeksforGeeks. "Supervised and Unsupervised learning." https://www.geeksforgeeks.org/supervised-unsupervised-learning/?ref=gcse (accessed February 7, 2022).

[35] P. Baja. "Reinforcement learning." https://www.geeksforgeeks.org/what-is-reinforcement-learning/ (accessed February 8, 2022).

[36] T. Branham. "6 Telltale Signs You Need Air Cylinder Repair." https://blog.wcbranham.com/6-signs-air-cylinder-repair (accessed April 24, 2022).

[37] W. Badr. "5 Ways to Detect Outliers/Anomalies That Every Data Scientist Should Know (Python Code)." https://towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623 (accessed April 26, 2022).

[38] A. A. Bajaj. "Time Series Prediction: How Is It Different From Other Machine Learning? [ML Engineer Explains]." https://neptune.ai/blog/time-series-prediction-vs-machine-learning (accessed May 12, 2022).

[39] Eddie_4072. "Feature Scaling Techniques in Python – A Complete Guide." https://www.analyticsvidhya.com/blog/2021/05/feature-scaling-techniques-in-python-a-complete-guide/ (accessed April 25, 2022).

[40] P. J. G. G. K. L. O. J. Oksum, "Case project 2 report, Testing different machine learning models in an oil well test rig for estimation of flow and fluid type," IIA1217-1 20H Hard/Soft Sensors in Process Measurement, At University College of Southeast Norway (USN), Tech repport 2020.

[41] F. Müller. "Getting Started with the Anaconda Python Environment for Machine Learning." https://www.relataly.com/anaconda-python-environment-machine-learning/1663/ (accessed April 25, 2022).

[42] A. Singh. "A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python code)." https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/ (accessed April 26, 2022).

[43] Q. K. T. S. A. Bayen, *Python Programming and Numerical Methods*, 1. ed. Elsevier Wordmark: Academic Press, 2020.

[44] F. Savvidou. "Connect to Azure Database for MySQL using Python." https://foteinisavvidou.azurewebsites.net/connect-to-azure-database-for-mysql-using-python/ (accessed April 21, 2022).

[45] NCD. "868MHz (Europe Only) Long-Range Wireless Mesh Communications Module." https://store.ncd.io/product/868mhz-long-range-wireless-mesh-communications-module-europe-only/ (accessed April 20, 2022).

[46] G. labs. "Alert notifications." https://grafana.com/docs/grafana/latest/alerting/old-alerting/notifications/ (accessed Mars 22, 2022).

[47] W. Duff. "What Is an SMTP Server?" https://sendgrid.com/blog/what-is-an-smtp-server/ (accessed April 19, 2022).

[48] A. Microsoft. "SQL Server on Azure Virtual Machines." https://azure.microsoft.com/en-us/services/virtual-machines/sql-server/#overview (accessed Mars 25, 2022).

[49] J. H. P. Savjani, A. Buck, M. Ghanayem, S. Gaur, H. James Toland III, Mel, D. Coulter, M. McCready, "Azure Database for MySQL Single Server." [Online]. Available: https://docs.microsoft.com/en-us/azure/mysql/single-server-overview

[50] J. H. P. Savjani, A. Buck et al., "Azure Database for MySQL - Flexible Server." [Online]. Available: https://docs.microsoft.com/en-us/azure/mysql/flexible-server/overview

[51] "MySQL Community Resources." https://dev.mysql.com/community/ (accessed Mars 24, 2022).

[52] R. Cobins. "An Introduction to Azure Cloud Security Features." https://securityboulevard.com/2022/03/an-introduction-to-azure-cloud-security-features/ (accessed April 15, 2022).

[53] S. M. M. G. P. S. kummanish. "SSL/TLS connectivity in Azure Database for MySQL." https://docs.microsoft.com/nb-no/azure/mysql/concepts-ssl-connection-security (accessed Mars 21, 2022).

[54] B. M. Elazar Krieger, Mel. "What is Microsoft Defender for Cloud?" https://docs.microsoft.com/en-us/azure/defender-for-cloud/defender-for-cloud-introduction (accessed Mars 28, 2022).

[55] A. A. S. Wray. "Backup and disaster recovery for Azure applications." https://docs.microsoft.com/en-us/azure/architecture/framework/resiliency/backup-and-recovery (accessed April 6, 2022).

[56] P. S. R. L. M. G. K. S. D. C. s. A. L. J. H. k. R. A. j.-e. N. S. H. J. Toland. "Backup and restore in Azure Database for MySQL." https://docs.microsoft.com/nb-no/azure/mysql/concepts-backup (accessed April 6, 2022).

[57] A. Joshi. "Learn how to manage security in PostgreSQL [Tutorial]." https://hub.packtpub.com/learn-how-to-manage-security-in-postgresql-tutorial/ (accessed February 12, 2022).

[58] Node-RED. "node-red-contrib-bcrypt." https://flows.nodered.org/node/node-red-contrib-bcrypt (accessed April 7, 2022).

[59] B. G. L. Team. "Create users and teams." https://grafana.com/tutorials/create-users-and-teams/ (accessed Mars 22, 2022).

Appendices

Appendix A: Master's Thesis task description.

Appendix B: IIOT Gateway WizX presentation.

Appendix C: SQL syntax used in the local database and SQL Script for database tables used in Microsoft Azure for MySQL Server.

Appendix D: Application Packaging Machine Sequences Diagrams (SDs).

Appendix E: Application Packaging Machine dashboard screenshots.

Appendix F: Node-RED flow screenshots.

Appendix G: Application PM test case document.

Appendix H: Application PM screenshots during the testing.