CIMAC CONGRESS 25
ZURICH, MAY 19 – 23, 2025

2025 | 297

Digitalization in filtration – how data-driven maintenance can help protect your assets

Digitalizatiion, Connectivity, Artificial Intelligence & Cyber Security

Andreas Schilbert, Boll & Kirch Filterbau GmbH

ABSTRACT

At a time when the maritime industry is increasingly reliant on advanced technological solutions, this paper explores the critical importance of maintenance in optimizing the performance of filtration systems with a filtration fineness of less than 10 microns - a key innovation in protecting marine engines and vital onboard mechanisms. As a filtration company integrating digitalization, connectivity and the Internet of Things (IoT) into its cutting-edge products, we emphasize the need for careful maintenance protocols to ensure the longevity and effectiveness of these fine filtration solutions.

Our discussion highlights how IoT-enabled sensors embedded in the filtration systems enable real-time monitoring of key parameters, including filter saturation and particle accumulation, which are particularly important for maintaining the integrity of sub-10 micron filters. The paper highlights the effectiveness of these advanced filtration technologies through a case study on land, draws parallels with their applications at sea and demonstrates the central role of regular maintenance to prevent performance degradation over time.

Given the enhanced susceptibility of finer filters to clogging and the consequential impact on engine performance, we stress that proactive, data-driven maintenance is not merely beneficial but essential. We created an in-house online platform which enables our service providers to identify trends in filtration performance and act on that information immediately. We use key performance indicators and our knowledge from performance tests within the research and development department to correctly assess the state of our filters and issue automated alarms to the users if the performance is degrading. This way we can transition from scheduled to condition-based maintenance and guarantee filtration performance through the whole lifecycle of our product. While we are still using humans in the loop to ensure a satisfactory customer experience, we are also exploring the possibilities of automating the process based on anomaly detection with more advanced machine learning algorithms.

1 INTRODUCTION

This paper describes the currently deployed data driven maintenance system at the Boll & Kirch Filterbau GmbH. The system integrates commonly used open-source software with a custom-made user frontend for internal and external users.

The reason for developing a platform for our filtration solutions is the extension of filter element lifetime. With increasing filtration fineness and demand for continuous operation the need for condition-based maintenance is even more pressing than before. Unplanned downtime can be costly and must be prevented.

The current solution for all Boll & Kirch Filters equipped with control boxes consists of counters for flushing events and filter runtime which are only accessible locally at the filter. No alarms or reminders are issued to the user when a service is due. The counters are manually checked and compared against maintenance intervals taken from the filter's manual. Filter performance alarms are based on a differential pressure indicator with two level switches signalling 75% contamination to trigger a backflush or 100% contamination to trigger an alarm. Alarms are only locally signalled by a red LED or can be relayed from the control box by using a digital output. There are no further measures in place to indicate a degradation of filter performance or filter element lifetime.

In a first step we developed a cloud-based monitoring system. Cloud-based systems are easier to iterate on and advance the solution faster [1]. A centralized approach also makes it easier to collect sensor data and perform explorational data analysis and test algorithms. This paper introduces the cloud architecture we use to collect data, the security measures in place to prevent unauthorized access and the algorithms used to analyse incoming data. The next step as described in the outlook will be bringing our algorithms from the cloud back to the control boxes and therefor not depend on a continuous connection to the internet which is detrimental to some filtration use cases.

2 ARCHITECTURE OVERVIEW

The architecture we use is based on a reference architecture provided by Microsoft [2]. Figure 1 shows an overview of the whole system with its subsystems. Our control boxes (1) send data to a broker (2) which handles authentication and authorization. The data is then forwarded to a time series database (3) for storage. Our infrastructure uses built in database functions to check for alarms on the data. The application BLUEtwin (4) visualizes the data and sends out alarms to users via mail.

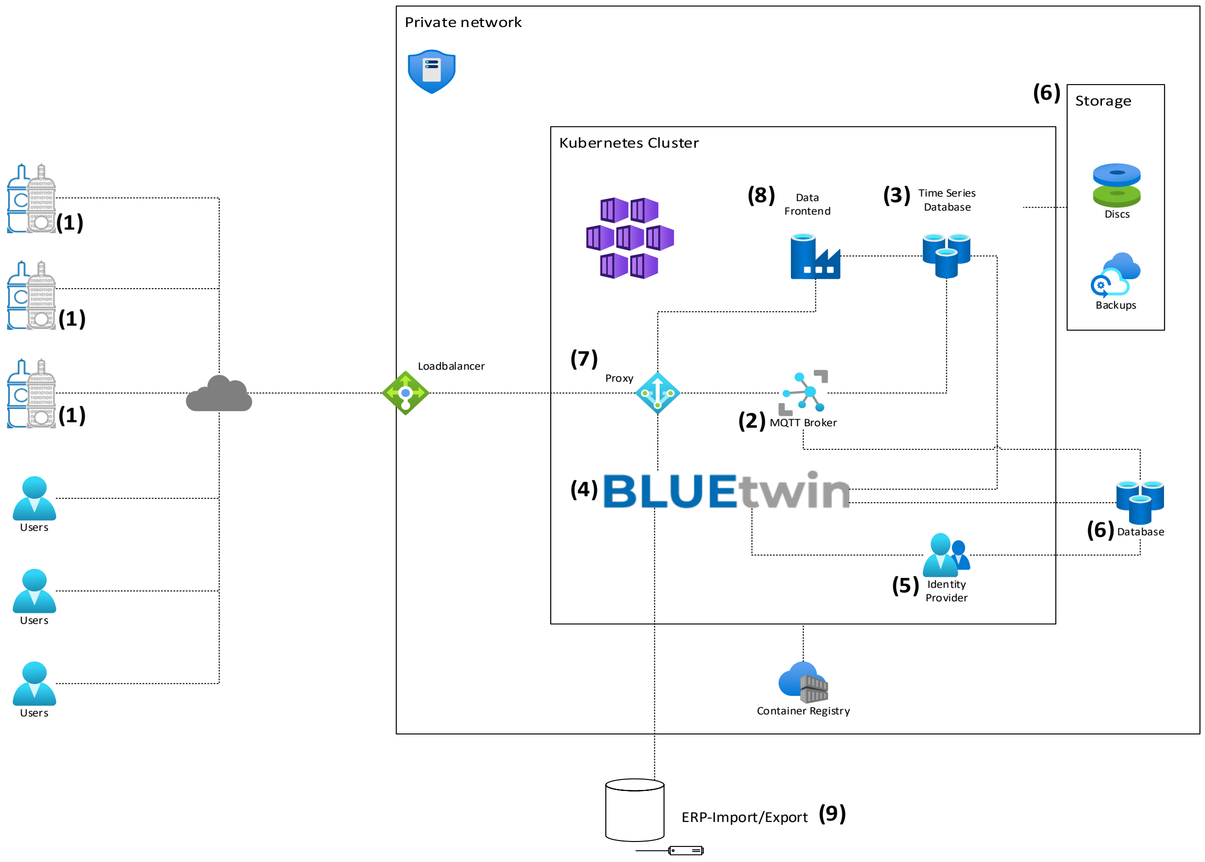

Figure 1: Architecture

2.1 Communication

The protocol we use for communication between the filter control box and the cloud is MQTTs. It is a lightweight messaging protocol for IoT usage which works with limited bandwidth. The transmission is encrypted via TLS. A broker (2) is hosted in our cloud environment which authenticates clients and applies access control lists to registered devices. MQTT is a publish/subscribe based protocol. Access control lists make sure that authenticated devices only publish or subscribe to those topics they have access to.

2.2 Data storage

Time series data is forwarded and stored in an optimized database (3). This kind of database is required when storing sensor data and comes with built-in analytic functions. We later use this analytic functions to perform data transformation and generate insights for our users.

2.3 Bluetwin

The application (4) which the user connects to. It is tailor-made for our needs and hosts the frontend to our data. It gives the users key performance indicators for his filters and gives recommendations on actions to be taken.

2.4 Further components

There are more components required to run our service which are used for user management (5), persistent storage of data (6), internally routing traffic (7) and manually inspecting data (8). We also connect to our ERP system (9) to import data like user manuals.

3 SECURITY

To identify security threats, the STRIDE model by Microsoft was used [3]. STRIDE is an acronym for spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privileges. For every connection between systems these threats must be evaluated, and appropriate measures must be taken. In the following section the security measurements of our control box are described.

3.1 Control Box

The primary goal is to always keep the filter operational. A first security measure is separating the IoT functions from the filtration functions on the hardware level. This means initiating a flush based on input from sensors is handled by a separate microcontroller (filter controller) which is unchanged from previous generations of control boxes. A second microcontroller (IoT controller) is used to collect the data over a serial connection to the filter controller, temporarily store it and send it to the cloud as soon as an internet connection is available. This makes sure that denial of service attacks, tampering or elevation of privilege attacks from the internet will not have any effect on the filter performance. Physical access to the device is the only way to influence the filters functionality.

The connection between the IoT controller and the broker is secured by using TLS encryption, authentication, authorization and ensuring availability of over the air updates to the firmware. To combat spoofing the broker is using certificates signed by a certificate authority. During the TLS handshake the IoT controller checks for the correct server certificate and only establishes a connection if it can be verified against a root certificate issued by the certificate authority. TLS encryption also secures the connection from tampering any data send by the IoT controller. The broker only accepts the connection after authenticating the control box credentials and ensures the connected control box only publishes information to authorized topics by applying access control lists. Reacting to new threats is only possible by constantly updating the running firmware. The IoT controller continuously checks for updates and installs the latest signed firmware. The signature prevents an attacker from installing firmware and therefor taking over the IoT controller.

4 DATA ANALYSIS

For this paper it is mandatory that the filter is equipped with a differential pressure sensor and connected to the internet. The IoT controller can store a limited amount of data in case the internet connection drops. After one day up to a week, depending on the number of sensors and the logging interval, the oldest data will be overwritten. As soon as the internet connection is reestablished stored data is sent to the cloud. The proposed method only applies to filters with differential pressure based flushing cycles, which means a flushing event is initiated when a certain differential pressure is reached.

The filter sends data periodically to the broker. If sudden events occur, which lead to rapid changes in sensor values, additional data is logged and send to the broker. Data we use to perform our analysis on is the differential pressure, the cumulative value of flushings as well as the cumulative value of operating hours.

The data analysis focuses on three main goals:

1. Usage based maintenance prediction.
2. Performance based maintenance recommendations.

3. Anomaly detection.

As described in the introduction there are currently no mechanisms in place to inform the user about upcoming maintenance and alarms are only issued by evaluating sensor data against a threshold.

By predicting the next maintenance based on actual filter usage we enable the user to shift from unplanned to planned maintenance. The next step is evaluating filter performance which enables us to dynamically give recommendations to ensure maximum filter lifetime. At last, detecting anomalies earlier gives the user time to react and avoid any potentially harmful operating conditions for the filter.

4.1 Maintenance prediction

For all supported filter series, the associated maintenance intervals for the wear sensitive parts are stored in a database. Those intervals are also found in the user manuals which are shipped with our products.

Maintenance intervals are purely based on cumulative flushings or operating hours, which can be confirmed by continuously running the filters on test benches. The filters are shipped into vastly different applications, which makes it hard to predict actual maintenance intervals in terms of months or years. To give a prediction on the next maintenance historical data of the specific filter is needed.

Our application uses the historical data of operating hours and flushings to calculate a usage-based estimation of the next maintenance date. The algorithm uses the historical data of a given timeframe and calculates an average filter usage for the given timeframe. Based on the predetermined maintenance intervals in the database it extrapolates the timeframe until the next wear limit is reached and informs the user about the date.

For a given filter and time interval Δt the amount of flushings f_{Start} and operating hours $OpHo_{Start}$ at the start of the interval and flushings f_{End} and operating hours $OpHo_{End}$ at the end of the interval are loaded from the time series database. Those are used to calculate the totals as described in (1) and (2).

$$f_{Total} = f_{End} - f_{Start} \quad (1)$$

$$OpHo_{Total} = OpHo_{End} - OpHo_{Start} \quad (2)$$

With these values the average flushings f_{Avg} and operating hours $OpHo_{Avg}$ for the given time interval are calculated.

$$f_{Avg} = \frac{f_{total}}{\Delta t} \quad (3)$$

$$OpHo_{Avg} = \frac{OpHo_{Total}}{\Delta t} \quad (4)$$

With the current flushing count and operating hours value the database is queried for the maintenance interval the filter is currently in and the corresponding limits are used for further calculations.

$$f_{Limit}: Limit\ of\ flushing\ count \quad (5)$$

$$OpHo_{Limit}: Limit\ of\ operating\ hours \quad (6)$$

The average filter usage combined with the limits can now be used to estimate the date of the next maintenance t_{Main} by comparing the next maintenance based on flushings cycles $t_{Main,f}$ and the next maintenance based on operating hours $t_{Main,OpHo}$.

$$t_{Main,f} = f(t, f_{Limit}, f_{Avg}) \quad (7)$$

$$t_{Main,OpHo} = f(t\ , OpHo_{Limit}, OpHo_{Avg}) \quad (8)$$

$$t_{Main} = min(t_{Main,f}, t_{Main,OpHo}) \quad (9)$$

The user is then informed about the result of the estimation in the front end of the application as shown in Figure 2. It shows the needed spare parts as well as which evaluation the estimation is based on.

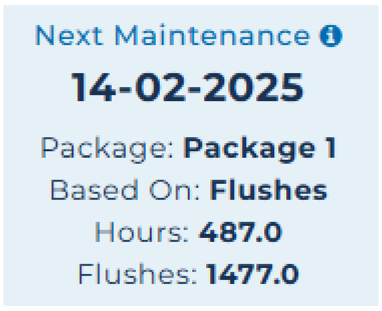
Next Maintenance
14-02-2025
Package: **Package 1**
Based On: **Flushes**
Hours: **487.0**
Flushes: **1477.0**

Figure 2: Front end view

4.2 Flushing performance trend analysis

To further extent the lifetime of our filtration solution we also analyse trends in the differential pressure data. Automatic filters are self cleaning and therefore do not need to be cleaned manually. Over time automatic filter elements also build a particle load which can not be cleaned by the backflushing system anymore. A manual cleaning can get rid of this base load and lead to increased flushing intervals and therefore an increased filter element lifetime since the backflush introduces additional stress into the element.

Catching a trend in backwash performance by manually reading values from gauges is nearly impossible. The most important indicator for a base contamination of the filter mesh is the differential pressure directly after a flushing event. With access to historical sensor data, we can calculate a trend line and not only visualize trends but also predict when the next manual cleaning is due.

To achieve this, the raw sensor data collected from our filters is used by searching for flushing events and calculating the differential pressure after a flushing event occurred. For every filter registered to our application we automatically save a baseline value for the post flushing differential pressure $dp_{Af,Base}$ when the first data is sent. For every Filter an alarm is created in the database which periodically checks if the current value is bigger than a limit value derived from the base value by multiplying with a factor k_1.

$$dp_{Af,Limit,clean} = k_1 * \ dp_{Af,Base} \qquad (10)$$

If the limit is reached the user is alarmed by the system and a recommendation to clean the element is given. The system also asks the user to inform us about the exact time of the manual cleaning which is then used to compare the after flushing differential pressure before and after the manual cleaning to judge its effectiveness.

A second limit is introduced for every element type which is used to give recommendations for changing elements because manual cleaning did not succeed. This limit is based on the element design and can not be changed by the user. If the differential pressure after flushing reaches this limit, we recommend changing the element completely.

To add a predictive part to the analysis we also apply time series forecasting by using exponential smoothing on the differential pressure after flushing. Figure 3 shows an exemplary time series of an automatic water filter used for desalination with a filtration degree of under 10 µm. The data is taken from a customer filter and spans thirteen days of operation. Oscillations to extremely low values of differential pressure after flushing are caused by forced flushing events when the filter is in an idle state. No volume flow is passing through the pressurized filter and the system initiates a flush based on a timer. While oscillations to higher values are caused by heavy particle load after which the filter needs several flushing cycles to recover. The differential pressure after flushing shows raw sensor data with no further smoothing applied.

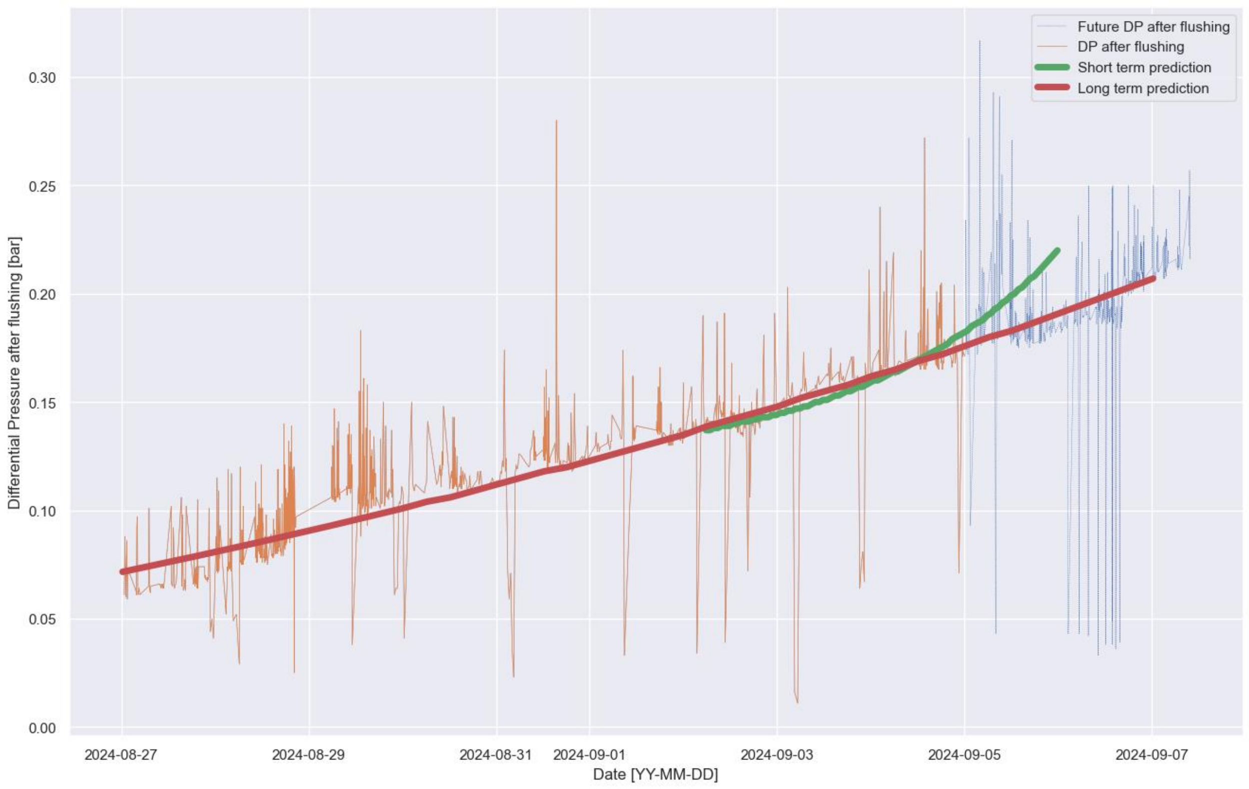

Figure 3: Trend analysis

The orange part of the time series consists of the known data the algorithm uses. The blue part are the future values not known to the algorithm and are only used to visualize the prediction error, since we are back testing the exponential smoothing on historical data in this example. The red line shows a prediction of the coming two days based on long term data (last fourteen days), while the green line shows a prediction of the next day based on short term data (last three days). By nature, the short-term prediction is much more volatile to changes in filter performance. Comparing the short and long-term predictions (green and red line) to the future values (blue line) we can observe that the algorithm is able to catch the trend in the data and give a good estimation of the differential pressure after flushing for the following days.

Both trends are added to the limit alarms and are used to inform the users about upcoming manual cleaning recommendations. Trends coupled with current data help the operator to make informed decisions on when to perform maintenance and prevent unplanned downtime.

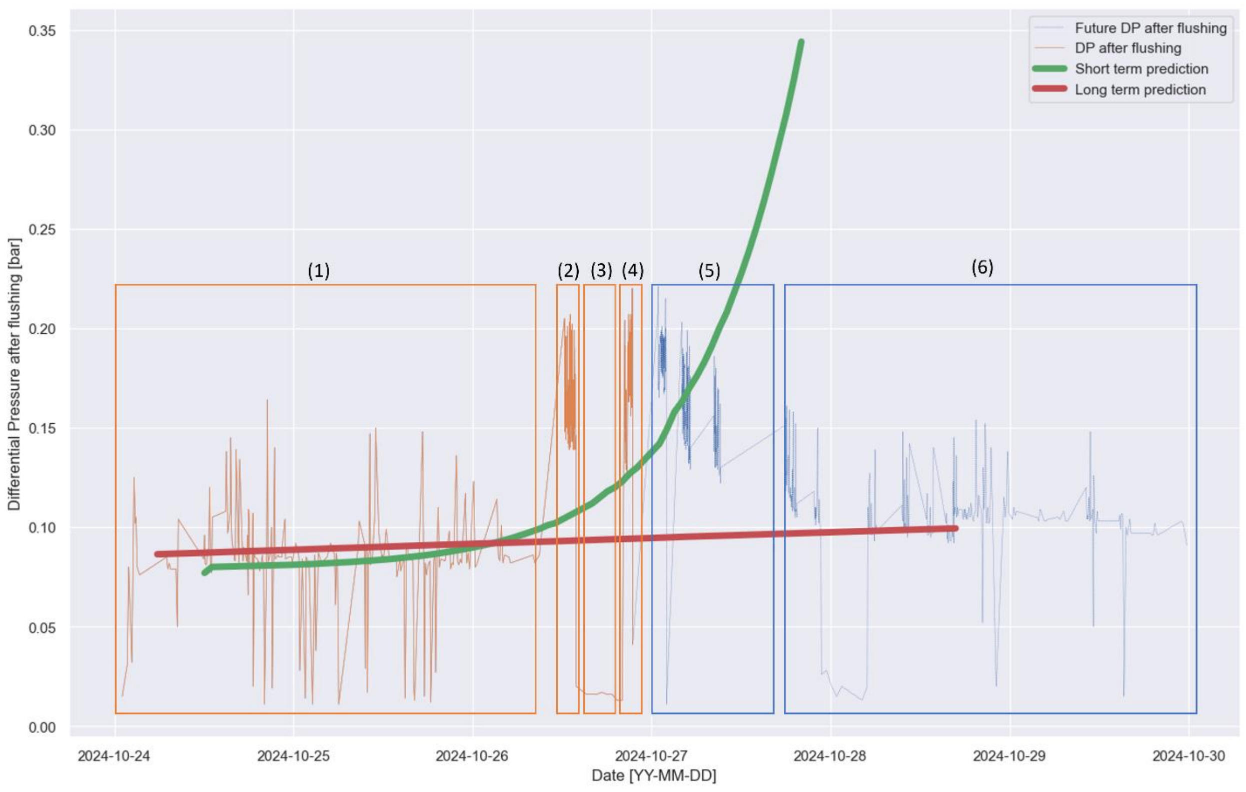

Figure 4: Example of short time heavy load operation condition

A limitation to this approach is shown in Figure 4. The filter is under normal load for approximately two and a half days (1). After that we see a sharp rise in differential pressure after flushing (2) most likely caused by a heavier particle load. Even a short period of no volume flow (3) with forced flushings by timer does not lead to a lower differential pressure after flushing in the next phase of operation (4). Over the course of another day (5) the load reduces, and the filter returns to normal operation (6). The short-term data (last three days) suggests a sharp rise in differential pressure after flushing as shown in phase (1) - (4). The prediction based on the short-term data (green line) acts accordingly and the user would have been informed about an upcoming cleaning recommendation while the long term data (last fourteen days) ignores the sudden rise in differential pressure and is in this case the better indicator, knowing how the filter reacted to the changing load. Since we operate on the differential pressure data alone, we do not know anything about the status of the overall system the filter is integrated in. Sudden changes in differential pressure can be caused by higher volume flows or an increased particle load. The final decision to perform a manual cleaning or full filter element replacement is therefor still in the hands of the operator. We try to remove any manual reoccurring and time-consuming tasks and only try to catch the operator’s attention when an action could be required.

4.3 Anomaly detection

To correctly interpret short term trends in the data we are currently developing a solution involving machine learning to correctly detect anomalies in our differential pressure data. This part is still in a prototyping phase of development and not used in a productive environment. We would still like to describe the current state of the solution to give an example of using machine learning for anomaly detection in time series data in a filtration context.

The model used on the data falls into the category of autoencoders. As the name suggests this family of models is used to encode and in a later stage decode the data to its original state again. Since these models are unsupervised, no labeled data is needed to train them. This contrasts with supervised learning models, which need hand labeled data to correctly learn connections between the input, for example images, and the ground truth, if the images show cats or dogs. The model is based on a convolutional neural network (CNN) described in [4]. It is fed a time series as input and reduces it to its main features in the encoding stage. When decoding the encoded data, it tries to reconstruct the given input based on the reduced features as an output. When using the model to detect anomalies we use this to our advantage. When decoding and encoding the times series data errors are introduced into the reconstruction. This reconstruction error can be used to find anomalies in the data. A low reconstruction error suggests a time series, which is similar to the one the model was trained on. When finding a time series with a bigger reconstruction error it is likely to be an anomaly. This approach also means that you only need non anomalous data to train your model.

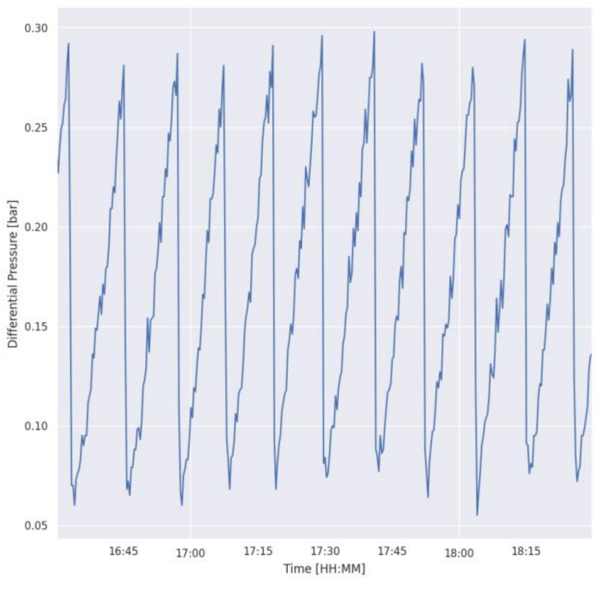

Figure 5: Input time series

During the process of training the model it tries to minimize the errors made when reconstructing the input. Figure 5 shows a part of the raw input time series data which is used to train the model. For training, the time series data is split into smaller series of 120 data points per series. A sliding window approach was used to create overlapping time windows of the mentioned size. To ensure no anomalies are present in the training data, 50 hours of filter operation were hand picked by an experienced engineer.

After training the model we can use it on our training data to infer the output data and visually evaluate the quality of the reconstruction. At this stage we want the model to reconstruct the input time series as closely as possible. Figure 6 shows an example of an original time series which was input into the model to infer its reconstruction. As you can see the reconstructed time series closely follows the original data. This is to be expected because the time series was part of the training data.

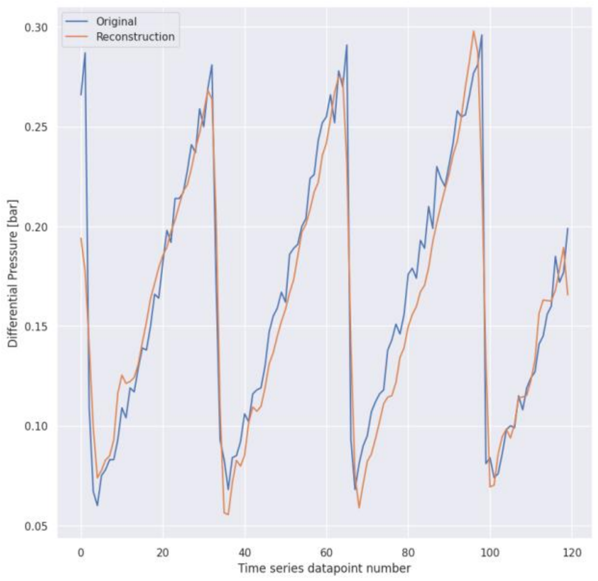

Figure 6: Reconstructed time series

To evaluate the general ability to find an anomaly which is easily spotted by an operator familiar with filtration, data is used which was not part of the training data and contains anomalies. The test sample was hand picked by the engineer again. The reconstruction error is expected to raise when evaluating an anomalous time series. A threshold ths is needed to judge if a time series belongs to the anomalous spectrum. This threshold is calculated by evaluating the reconstruction error in the training data set. The model input X, output Y and number of datapoints n in a training sample is used to calculate the mean absolute error MAE of every sample. The threshold is calculated by taking the average mean absolute error over all samples k and adding the standard variation σ to it.

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} |Y_i - X_i| \quad (14)$$

$$ths = \frac{1}{k} * \sum_{i=1}^{k} MAE_k + \sigma \quad (15)$$

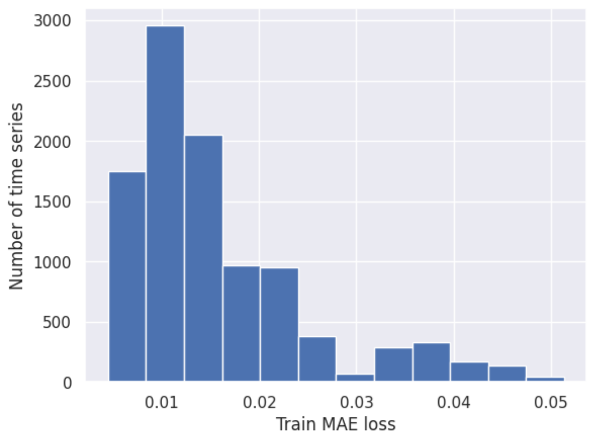

Figure 7: Training mean absolute error

Figure 7 shows the distribution of errors and the corresponding number of samples. Please note that the MAE ranges from 0 to 0.05 with a total number of training samples of 10100 series consisting of 120 datapoints.

The following step is using the trained model on unknown data to make a prediction or in our case a reconstruction. This step is called inference. Our test data example is taken from a field test of a water filter. When using the trained model on the test data we use the same data preprocessing approach also used on the training data. This means splitting the time series into smaller pieces of 120 data points using a sliding window. After applying the sliding window our test data consists of 7924 samples featuring 120 datapoints each. To evaluate if a time series is anomalous, we calculate the mean absolute error of the test data sample and apply the threshold from our training data. Figure 8 shows the distribution of the mean absolute error in the test data set. The calculated error scores are of a much wider range compared to the training data, which is to be expected since the test data contains known anomalies.

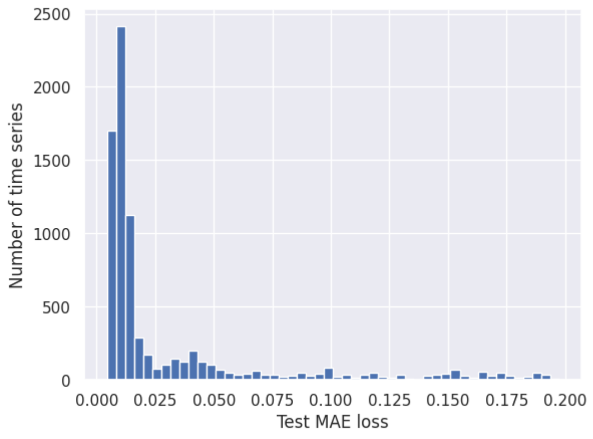

Figure 8: Test mean absolute error

Before looking at the detected anomalies we first check the reconstruction of a part of the time series data which was not flagged as anomalous. Figure 9 shows the original and reconstructed data of a sample from the test data. The model can reconstruct the input data, which means the sample is similar to the data the model was trained on and therefor not anomalous.

In contrast to that Figure 10 shows an anomalous time series sample from the test data. The model fails to reconstruct the original data and therefore this sample is classified as anomalous. From a filtration perspective the rising differential pressure after flushing coupled with an increasing flushing frequency can be observed in the original data (blue). This kind of behaviour was not present in the training data and is unknown to the model which leads to the failed reconstruction.

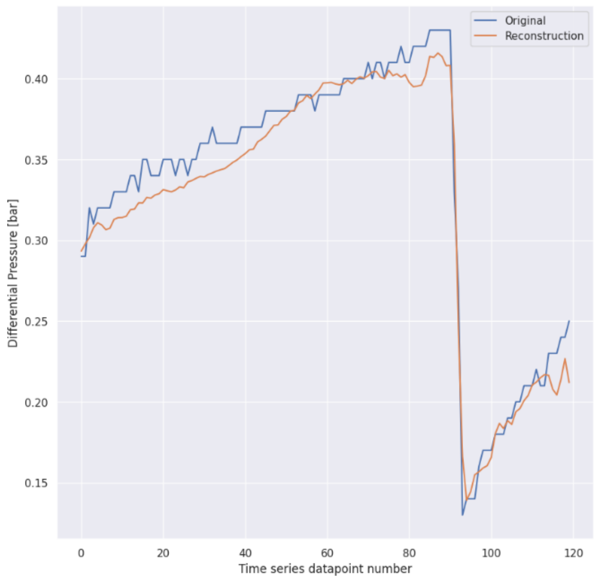

Figure 9: Reconstructed test time series

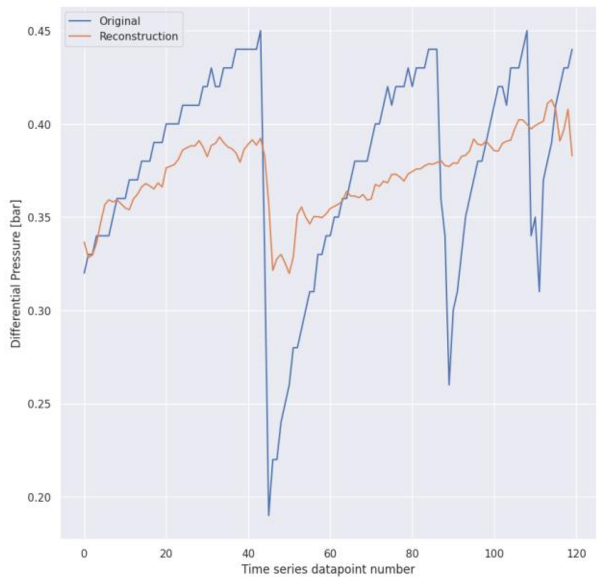

Figure 10: Anomalous sample test data

Figure 11 shows the complete example data from our test set. The data is taken from a test which led to a catastrophic failure. The model would have flagged the data as anomalous starting at 2:11 am. Our standard differential pressure indicator is set to initiate a flushing at 0.38 bar and issue an alarm at 0.5 bar. The differential pressure raised above 0.5 bar at 3:22 am. An alarm would have been issued about one hour earlier using the proposed model. By detecting the anomaly early, extra load onto the element is reduced, which occurs when a system is not shut down in time.

Analyzing the data from a filtration expert's perspective, one would have identified performance concerns regarding the filter at that time. The flushing intervals are getting shorter and the differential pressure after flushing events starts to rise. Combining these observations into a single actionable threshold was the goal when choosing the machine learning approach. Another advantage of autoencoders is not needing anomalous samples to train the model. Conventional signal analysis techniques rely on labeled data containing anomalies, which is not available to us in a sufficient amount. This is also the reason why no accuracy, precision, recall or a f1 score is presented. Further work is required to compile a meaningful test and validation dataset to ensure the robustness and generalizability of our model. As mentioned in the introduction to this section we do not use the model in production currently. It is only trained on a small sample of data and only applicable to automatic backwashing filters. We are currently evaluating how to implement a machine learning solution into our cloud architecture. Additionally, it is not clear if we can achieve satisfactory performance with only one model or if more models are needed based on filter series and application.

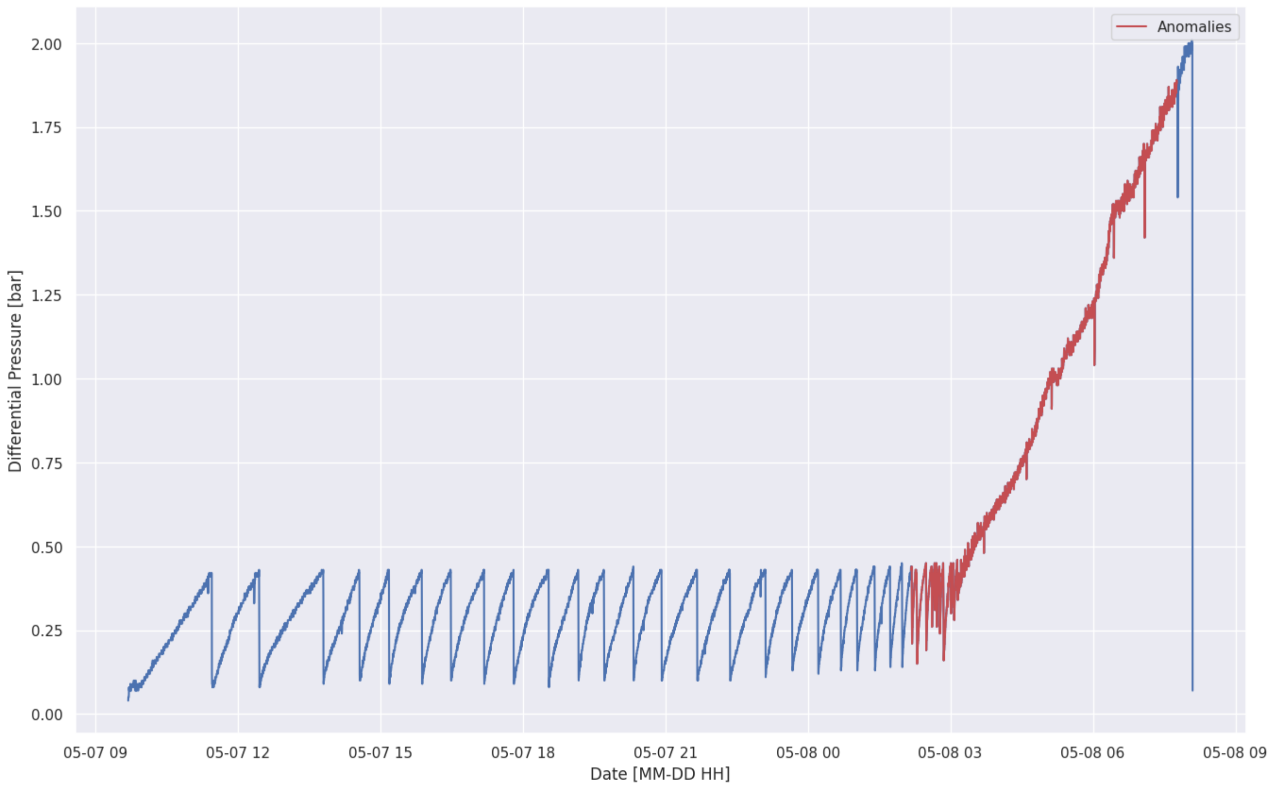

Figure 11: Complete test data time series

5 CONCLUSION

This paper has shown the complete architecture of a cloud-based data analysis system which is used to collect and store data in the cloud, analyse it and give feedback to the users.

By analysing historical filter usage, we can help our users schedule condition-based maintenance to reduce unplanned downtime. We no longer must rely on fixed maintenance intervals. Instead, we give a prediction on when the next maintenance must be planned according to actual usage a filter has seen.

We also have shown how to evaluate particle accumulation in our filter elements by evaluating differential pressure build up after flushing events in automatic backwashing filters. Coupled with trend analysis and forecasting we can give cleaning and replacement recommendations to ensure filtration performance over time.

Lastly, we had a look into a machine learning approach to automatically detect short term anomalies in our differential pressure data. Currently our filters would only alarm the user when the differential pressure rises above a predefined threshold. With the described approach we would be able to spot anomalies earlier and prevent unnecessary load onto our filter elements.

6 OUTLOOK

The approach taken in this paper has the disadvantage of needing a constant internet connection to work. We are aware that this is not possible in every scenario and much less in a maritime environment.

Our focus for the future is bringing the complete data analysis capability to our control box and therefore to the edge of the cloud. This will need a new generation of control hardware which is more capable than the current. We are working on a protype control box which can run a scaled down version of what is shown in Figure 1. The hardware we currently test runs a time series database as well as the machine learning model. We are positive a solution for industrial use in the same form factor as our current control box can be achieved.

7 DEFINITIONS, ACRONYMS, ABBREVIATIONS

DP: Differential pressure

MAE: Mean absolute error

8 REFERENCES AND BIBLIOGRAPHY

[1] AWS Cloud Economics, 2020. Cloud Value Benchmarking Study Quantifies the Benefits of Cloud Adoption: 10-11.

[2] Lynn, T., Mooney, J.G., Lee B. and Takako Endo, P. 2020. The Cloud-to-Thing Continuum Chapter 1: The Internet of Things: Definitions, Key Concepts, and Reference Architectures, 1st ed., Palgrave Macmillan Cham, Basingstoke, Hampshire, England.

[3] Kohnfelder, L. and Garg, P. 1999. The threats to our products. Microsoft.

[4] Stern, B. and Witschi, F. 2022. Machine Leaning-Based Analysis of Data from the ZHAW Movement Analysis Laboratory for Fatigue Detection during Sports Exercises. Züricher Hochschule für Angewandte Wissenschaften.

9 CONTACT

Andreas Schilbert, Senior Engineer Digital Innovation Lab R&D, Boll & Kirch Filterbau GmbH, andreas.schilbert@bollfilter.com