Artificial Intelligence for Human-Technologies-Economy Sustainable Development
23–25 May 2024
Lublin, Poland
make learn
TIIM
PIConf
Management, Knowledge and Learning
International Conference
Technology, Innovation and Industrial Management
Sustainable Development

Project Report - Supporting The Preventive Maintenance Plan for The Extruder On The EPDM Automotive Industry Production Line Using a Time Series Database

Łukasz Orzechowski
Lodz University of Technology, Lodz, Poland
e-mail lukasz.orzechowski@dokt.p.lodz.pl

Abstract

This Project report is a part of a bigger solution in the field of Real-Time Systems. An Informatic solution was created as a proof of concept in a project of SMART Extrusion EPDM in the company from the Automotive Industry. The problem that has been solved is providing real-time information to the maintenance team about the wear of the machine that extracts the EPDM rubber mixture. Replacing or repairing a machine is a high cost for a business that would like to have more precise information about the required expenses for the readiness of machines for production. The current situation did not allow for quick reporting and archiving of data in a digital way; reports could only be delivered on paper after a periodic maintenance review. The report proves that by adapting IoT solutions to industry requirements, it is possible to prepare a solution that is competitive with SCADA systems at a low cost. Companies investing in the Industry 4.0 concept using The Industrial Internet of Things (IIoT) have the advantages of simplifying, accelerating and reducing implementation costs by providing better, more modern analysis tools.

Keywords: time series database, big data, Internet of Things (IoT), database systems, data stream, InfluxDB, Message Queuing Telemetry Transport (MQTT), measurement system.

INTRODUCTION

Carrying out periodic inspections of the EPDM compound extruder is necessary to keep the machine in the best possible condition. Optimizing the maintenance process allows you to reduce unplanned downtime while enabling precise timing of planned downtime. An experienced maintenance team, using expert knowledge and the computerized maintenance management system reporting, is able to prepare an appropriate inspection or inspection schedule. The basic way to optimize inspections may be to rely on previous data from the machine's operating history. The key to streamlining and automating the delivery of necessary information is real-time monitoring of machine operation. The use of telemetry and data archiving, depending on the scale, may prove to be a big challenge for IT systems, as well as SQL databases. The use of IoT tools and time series databases in this field allows the preparation of reporting solutions in a shorter time while consuming fewer resources required for server operation.

In a dedicated monitoring system, the key element to be controlled is the rotation of the screw that moves the raw material towards the extrusion head. The wearable elements here are the screw and the cylinder. However, the wear and tear of the screw will be to a greater extent due to the material, design and method of operation. The rate of wear of the elements depends on factors such as the type of EPDM rubber mixture used and the speed at which the screw rotates during the work cycle. All the factors mentioned that influence consumption are determined by the characteristics of the finished product and are determined in the parameter card. It is very diverse, which makes it difficult to precisely determine the dates of required replacement or periodic regeneration.

Very big potential and low cost of solution MQTT, InfluxDB and Grafana compared to SCADA systems is a natural alternative of implementation. Putting aside the advantages of solutions derived from IoT, we can be sure of the success of an implementation based on such an alternative to traditional SCADA applications. Without a doubt, by integrating various physical components into information systems and connecting them to digital representations, the Internet of Things (IoT) idea broadens the scope of the traditional SCADA concept. This allows for the real-time collection and transfer of data across numerous protocols. [1]. We can admit that this sentence shows the way how IoT is evaluated and what was the idea behind all technology. Based on our knowledge of various communication protocols, such as CoAP, MQTT, AMQP, JMS, REST, and XMPP, as well as the realization that storing a significant amount of data in a relational database would cause query times to lag and incur significant storage costs, we have chosen to propose a solution that exploits MQTT as a lightweight communication protocol and a NoSQL database, called InfluxDB, for data storage.[2]. The performance of the database in keeping all the load is most important. It is therefore confirmed that the technologies implemented in the time series databases are a response to the need to implement solutions resulting from the need to analyse data over time. By using the MQTT protocol as a tool for distributed measurement communication, as well as Node-RED, an innovative programming approach, to process data in microprocessor systems, and also time series database systems like InfluxDB, we can cover the process of ETL for most applications. Putting on the top layer data visualization tools like Grafana, we gain a system for most applications to flexible reporting solutions [3]. InfluxDB is a complete platform that facilitates time series data gathering, archiving, monitoring, visualization, and alerting. unfortunately, most of the components, apart from the time series database engine itself, are still tools that only try to catch up with the possibilities of solutions from Grafana Labs. It is possible to create a solution solely based on the InfluxData ecosystem because it is an open-source platform and is constantly improved not only by the founders but also by the community that uses the solutions in their projects. Stack the

following four fields: Telegraph, InfluxDB, Chronograph, and Kapacitor give tools to cover very specialized roles, such as data storage, time series visualization, metrics collection, and predefined post-processing procedures [3]. Paying attention only to the database engine of well-known relational databases and all available noSQL databases such as KdB+, Prometheus, Graphite, or TimescaleDB, we can certainly say that InfluxDB is one of the most popular native time series databases [4].

MATERIALS AND METHODS

The applicable procedure in the maintenance plan includes steps related to organoleptic control of the extruder in periods of not less than a week. The control of element wear is based on measuring the thickness of the screw blades (B) and the walls of the cylinder (A). The data is archived and stored in control sheets. An experienced Maintenance expert calculates periodic wear and tear and then, based on his expert knowledge, decides on the need for regeneration or replacement. The cost of replacing an element corresponds to the cost of regenerating it four times. Systematic inspections and measurements

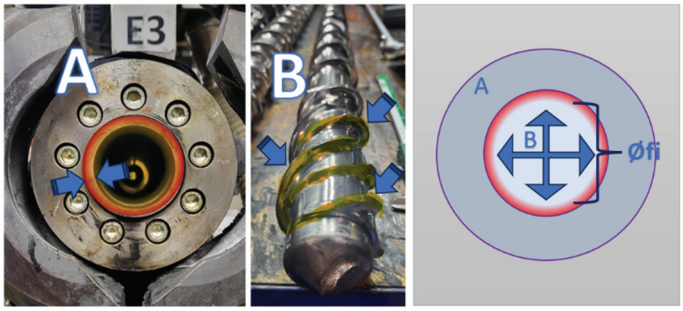


Figure 1. Cylinder and screw with an indication of the wear areas.

of component wear play an important role in such a plan. However, it happens that maintenance plans become a low priority and Maintenance teams are redirected to other tasks. Due to the lack of systematic measurements, machine wear control sheets may not reflect the current condition and therefore do not provide the information necessary to prepare automatic detection methods. In order to accelerate measurements and provide information in real time, a team of Maintenance experts selected a key parameter, which is the number of revolutions of the extruder screw. The main assumption here is that the automation solution will be able to count turnovers and send them to the database in real time. The parameters to be monitored are the set revolutions (Set Point - SP) and the read revolutions (Current Value - CV). The parameters will be saved in revolutions per minute (rpm). Data can be read and transferred with a frequency of 100 ms. During the analysis, a reading frequency of 1000 ms was finally selected, i.e. a minimum of 60 times per minute. The Extract Transform Load (ETL) solution is based on communication with a Siemens industrial PLC controller, and the reading and communication protocol is Step7 as the default for this manufacturer. The transport layer for communication is the TCP/IP protocol and a standard LAN network. The entire ETL process should be divided into several steps located in three areas. Due to the strict requirements of the standards that PLC controllers meet, one of the first areas that had to be isolated was the industrial network. This area is a separate segment of the industrial network dedicated to the communication of all machines and devices of the production line. It is a kind of autonomous zone managed and developed by experts in the field of automation. For security reasons, direct communication between the database server and the controller is not possible. An edge device should be used that provides the ability to acquire data and create a network. A Data Acquisition Device (DAD) is primarily a highly specialized, reliable and secure edge device that also performs the tasks of translating protocols such as OPC UA, Step7 or ModBUS to IoT standards,

including MQTT. DAD must be configured to work with an industrial network and a network dedicated to the Internet of Things, created for security purposes. Therefore, the IoT network is another separate area isolated from the corporate network by a hardware firewall. The allowed traffic between the IoT network and the corporate network is limited only to communication with the database server on the necessary TCP/IP communication ports. The last area is the server network and the user network. This division is illustrated in the flow diagram (Fig. 2).

The use of such a division ensures non-negotiable, broadly understood digital security, absolutely

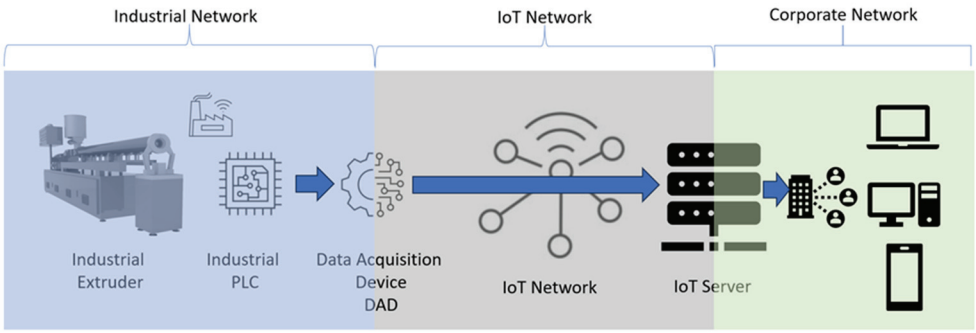


Figure 2. Data flow diagram in the ETL procedure.

required in the industry. Another advantage is scalability for large production environments, where devices from one production line are isolated from each other. The created hardware architecture is able to meet the IATF 1 Preparing the physical layer of the solution requires knowledge of communication and protocols used in PLC controllers. Very good knowledge of computer networks is also required. A relatively new area here are IoT issues, i.e. device configuration, MQTT communication protocols, server and databases. The extensive MQTT standard requires setting up the services and database layers in order to be used. A review of available solutions ensuring the ability to handle information resulting from the use of the MQTT protocol required the selection of services such as:

- MQTT broker – a service through which clients connect to exchange information;
- Event Driven Application (EDA) – an application that allows you to create flows;
- Time series database – a database optimized for reading and writing time series data;
- Dashboard platform – a specialized platform for interactive visualization.

The number of solutions used may seem complicated, but building such an infrastructure offers many possibilities and is perfectly scalable for a wide range of applications. Each of the above-mentioned elements protects a different technical area of the entire solution. Thanks to such advantages as the lightness and simplicity of the MQTT communication protocol operating on the upper layer of the

TOPIC MQTT Schema:
Plant/Line/ProductionMachine/MachinePart/Name of parameter/Type/Units

example
Plant A/Extrusion Line 8/Extruder/Driver/Screw Rotation/CuretnValue/RMP

Figure 3 Scheme of MQTT TOPIC format.

TCP/IP protocol, we gain an efficient way of transmitting information. The MQTT protocol works based

on the publish/subscribe model, which enables asynchronous communication between publishing and subscribing clients. In the designed solution, the publishing client is a DAD edge device, and the subscriber is an IoT server with the MQTT Broker service. Each MQTT message minimally consists of a TOPIC and a PAYLOAD of transmitted data. An important element is to prepare a schema for the MQTT message so as to easily identify messages on the broker's side. One of the ways is to format a topic schema that can identify the machine and the measurement. For greater scalability, TOPIC may additionally consist of an identifier of a production plant, production line, machine, or machine part. Each transmits cycle to the database will be initiated every 1 second. The process starts and is controlled by the DAD device. In the first step, reading is performed from the SIEMENS controller via the STEP

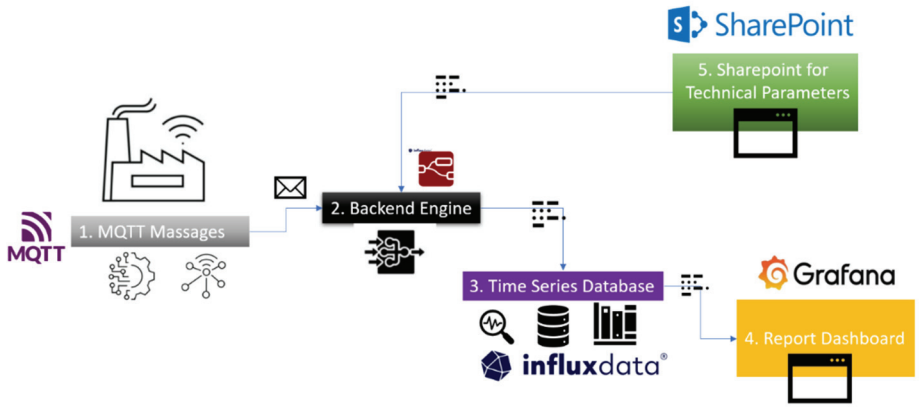


Figure 4. Information transition lifecycle.

7 communication protocol. Then, thanks to the use of a translation table and a processing program, the information is transformed into an MQTT message and sent from the DAD device (Figure 4-1) to the MQTT broker. The message sent to the broker, passing through a separate VLAN, goes via FIREWALL to the MQTT server. On the MQTT server, the running Broker service intercepts the message. At this stage, the MQTT broker should be imagined as a service listening on a selected TCP port, not implementing any business logic. Messages reaching the Broker are queued and stored according to delivery priority in the server's memory. At this stage, you can connect to the Broker in subscriber mode and wait for the next message according to the TOPIC header we are interested in. A node prepared in this way allows the message to be used by the TELEGRAF database engine to immediately save information to a Time Series database (Figure 4-3) or to intercept this message by Event Driven Application (EDA) in order to implement additional actions or dedicated processing that fits into the logic of the real-time system. EDA (Figure 4-2) is a backend engine enabling the implementation of flows in a very wide range. The goal is to store and analyze data, which is why each message sent is saved to the Time Series Database (Figure 4-3), a specialized engine that perfectly copes with the collection and processing of data that changes quickly in time.

InfluxDB is a time series database designed to handle large volumes of data with high upload/download rates. The basic data structure in InfluxDB is "measurement". Time series data supports multi-valued models. Measurement is a collection of data points that share a common set of

InfluxDB entity	SQL DB entity
Measurement	Tables
Tags	Columns indexed
Fields	Columns not indexed
Points	Rows

Figure 5. Compare SQL and InfluxDB entity

tags, fields, and timestamps. Tags are key-value pairs used to identify and organize data points, while fields are the actual data values associated with each data point. Timestamps are used to indicate when data was recorded. Each measurement is further divided into "points". A point represents a single data record and consists of a timestamp, a set of tags, and one or more fields. Timestamp is a Unix timestamp that represents the number of seconds since January 1, 1970 UTC. Tags and fields are key-value pairs stored as strings or numbers. In addition to measurements and points, InfluxDB also has a "database" concept. A database is a container for one or more measurements and is used to organize and manage data. Their measurements are organized into series, such as Map[key] = Series, where the key is a set of measurements plus a set of tags. The value of the InfluxDB measurement field can be saved as string, float, int or Boolean and is recognized automatically by the engine after the first saved point. Each measurement can have multiple tag sets and multiple field sets.

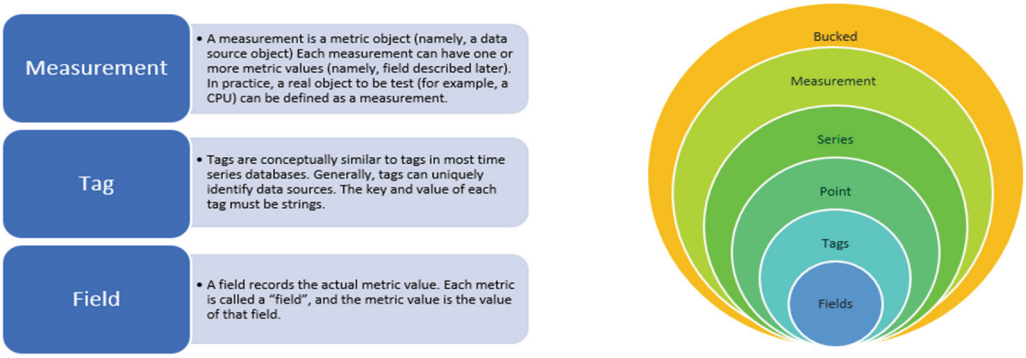


Figure 6. InfluxDB database concept diagram.

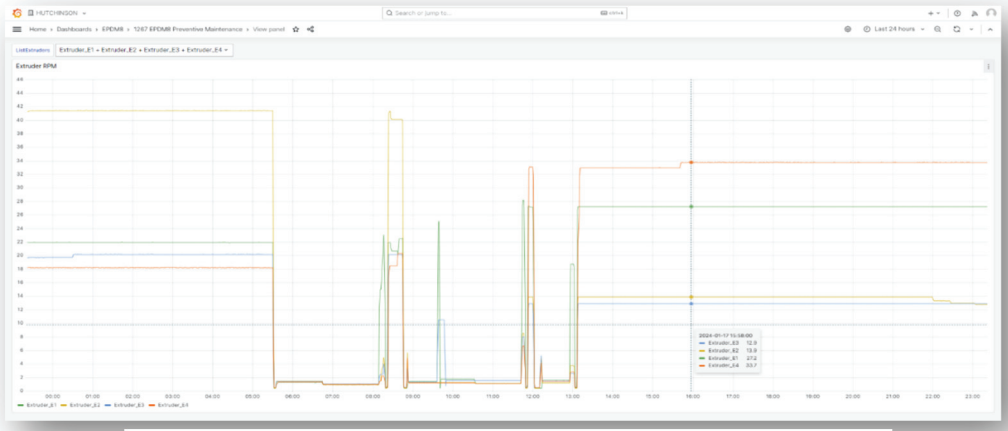
The launch and configuration of the InfluxDB service can be performed on the same server as the remaining elements of the entire solution. There is also no problem in preparing the service on a completely different machine, which ensures scalability of the service in the event of problems with the performance of the entire solution. In the variant in which the IoT server will have all the services, the key parameters will be RAM and the number of processor cores. In the distributed model, each service can be installed on a separate machine with an individual RAM configuration and number of cores. However, the bottleneck will be network communication between services. In this model, to avoid a lack of work continuity, stable and redundant network access must be ensured for each node.

Using the specific structure of the InfluxDB database, it is necessary to consider the scheme in which data flowing from the production line will be stored. In the described case, the "Measurement" measurement will be the rotation of the extruder screw. Additional tags will also be required to identify the plant, production line, machine and store information about the unit from which the data is loaded. A consistently selected TOPIC MQTT schema will be reflected in the tag, value, and fields association of the time series database schema. In the mapping function, the tag assignments are Plant, Line,

ProductionMachine, MachinePart, Unit, and Type. However, the required measurement field will be completed with the name of the Parameter Name. The schema created in this way will allow you to obtain a storage point in the database that can be isolated based on the most important attributes, thus providing the possibility of aggregation and filtering in database queries. The innovative technique of storing data in the InfluxDB database and the indexing mechanisms that the engine processes by default allow, on the one hand, to save data space and ensure data reading in an extremely efficient way. We have two ways to save data to the database. The first is an efficient Web API that can be implemented quickly in many programming languages. An additional native solution is Line Protocol, which can be used by setting the Telegraf agent, providing even greater recording performance. Each of the solutions provided is sufficient for the needs of the project without requiring a strong server configuration.

To summarize, the data processing process starts with reading a memory data block in the SIEMENS controller using an edge device. The read data is transferred in the MQTT format to the IoT server and then saved to the time series database. Each extrusion line has a minimum of 4 extruders. The project requires readings of two extruder rotation parameters: the set value SP (Set Point) and the read value CV (Current Value), which gives us almost 700,000 entries to the database every day. The data horizon needed for analysis is counted in months. Data visualization is provided thanks to the Grafana platform with connectors to the InfuxDB database. Within the InfluxDB database ecosystem, of course, there are also tools for visualization and data processing. However, a definitely better and more flexible choice is to use visualization panels in Grafana. The platform is installed as a service and allows access in the form of a WEB application to create interactive visualizations. Thanks to built-in connectors to the InfluxDB database and dozens of default types of visualization panels, as well as a constantly growing user community, Grafana is an excellent tool for data visualization that does not require programming. Data can be provided for analysis and visualization using the InfluxQL language, which is similar to the SQL standard, i.e., partially sharing the syntax. It is available in the first releases of InfluxDB, and its natural evolution has become the FLUX language, which has become the default form of querying in the latest versions of the engine. FLUX is more than just a query language; it is a full-fledged scripting language that can perform all kinds of complex transformations and analyses of time series data; we can classify it more into a group of NoSQL languages. Once we have collected the data, we can move on to analysis and visualization.

The goal is to create one universal query counting the extruder revolutions for any period between the start date and the stop date. Looking from the perspective of established SQL tools such as MsSQL RS (Microsoft SQL Server Reporting Services), visualization would most likely require additional steps in the form of tasks with a fixed time interval, permanently writing the aggregation results to the database. Traditionally, it would be necessary to prepare daily reports on a daily basis, additionally ensuring data redundancy, which would not ensure freedom of viewing in time anyway. So, let's start with the basics and prepare a simple visualization showing a turnover graph on the time axis. The query presented below is executed with a start and stop date set to 24 hours. The time to complete and refresh the visualization on the chart is 0.2 seconds. The same range query on day 7 has an execution time with a visualization of 0.28 seconds. A script created in FLUX starts by specifying the database name and then the time range. We can notice that each new line begins with the symbol "|>"; this is a command to forward the data stream to the next step. The data is filtered by tag, and then the result is passed to the aggregation function. The aggregateWindow function is a key performance element here because it samples data, groups it into fixed-time windows, and applies an aggregation function to each window. The time window is calculated before the query is executed based on the start and stop date range. In the case of a 24-hour window period, the engine calculates it to 4 minutes, while for 7 days, it will be 24 minutes. It is always possible to set a constant value of the time window, and in the presented case, it is 1 minute and visualize data for 24 hours. This query will be processed by the database 30-40 per cent longer, which is due to the number of points that need to be visualized on the chart.



```
from(bucket: "Plant A")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "ScrewRotation")
  |> filter(fn: (r) => r["ProductionMachine"] == "Extrision Line 8")
  |> filter(fn: (r) => r["TypeValue"] == "CV" or r["TypeValue"] == "SP")
  |> aggregateWindow(every: v.windowPeriod, fn: last, createEmpty: false)
  |> yield(name: "last")
```

Figure 7. Time history visualization of extruder rotation; FLUX query used to display extruder rotation.

The graph presented Figure 7 is a visualization of revolutions over time and you can see any operating anomalies as well as the ratio of the set speed to the read speed. This is therefore an excellent starting point for the Maintenance team, allowing for daily analysis of the device's operation. However, it is possible to prepare the query in such a way that it presents the total turnover for any period. This requires adding a few extra elements to the data stream. Let's pay attention to the range clause, which has been supplemented with a mechanism for cutting off and rounding the date range to the full minute. This is not very important, but it allows you to prepare the range more precisely, even when the user selects the time without paying attention to hundredths of a second. The data filter by tags does not change; it only requires narrowing down to the read value (CV) and omitting the set value (SP). The filter should also take into account the rejection of the extruder's operating range when it is idle, i.e. when the revolutions are below 1 rpm. The stream prepared in this way can be fed into the aggregation function in which the time window parameter of 60 seconds has been selected, not without significance. The prepared manual time window calculates the average number of revolutions within 1 minute. The final step is to add up the turnover value. The sum of the average per minute is the result closest to reality, which we can treat as an indicator of extruder screw wear.

```
import "date"
from(bucket: "Plant A")
  |> range(start: date.truncate(t: v.timeRangeStart, unit:1m), stop:date.truncate(t:v.timeRangeStop, unit:1m))
  |> filter(fn: (r) => r["_measurement"] == " ScrewRotation")
  |> filter(fn: (r) => r["Production_machine"] == " Extrision Line 8")
  |> filter(fn: (r) => r["TypeValue"] == "CV")
  |> filter(fn: (r) => r["_value"] >= 1)
  |> aggregateWindow(every: 60s, fn: mean, createEmpty: false)
  |> stateDuration(fn: (r) => r._value > 1, unit:1m)
  |> drop(columns: ["host", "topic"])
  |> count()
```

Figure 8. FLUX query showing the counted extruder revolutions

In order to complete the task, data on the extruder's operating time should also be presented using the same data. This task is made easier by the stateDuration function, which returns the accumulated duration of a given state. In our case, this is the operating status of the extruder, which is not idle. The function will return the time between the change from idle speed to the speed included in the working time. The runtime can be shown here as the sum of the minutes provided by the result of the stateDuration function. However, a better result was to count the number of rows returned by this function, assuming that the unit was 1 minute.

```
import "date"
from(bucket: "Plant A")
  |> range(start: date.truncate(t: v.timeRangeStart, unit:1m), stop: date.truncate(t:v.timeRangeStop, unit:1m))
  |> filter(fn: (r) => r["_measurement"] == "ScrewRotation")
  |> filter(fn: (r) => r["Production_machine"] == “Extrision Line 8")
  |> filter(fn: (r) => r["TypeValue"] == "CV")
  |> filter(fn: (r) => r["_value"] >= 1)
  |> aggregateWindow(every: 60s, fn: mean, createEmpty: false)
  |> drop(columns: ["host", "topic"])
  |> sum()
  |> yield(name: "sum")
```

Figure 9. FLUX language query presenting the counted extruder operating time

RESULTS

The query prepared in this way can be presented on charts or gauges. The gauges form was accepted by users as more illustrative and tailored to the needs of use in industry. Once exists the ability to count the revolutions for each extruder, it is worth taking care of the possibility of parameterizing the

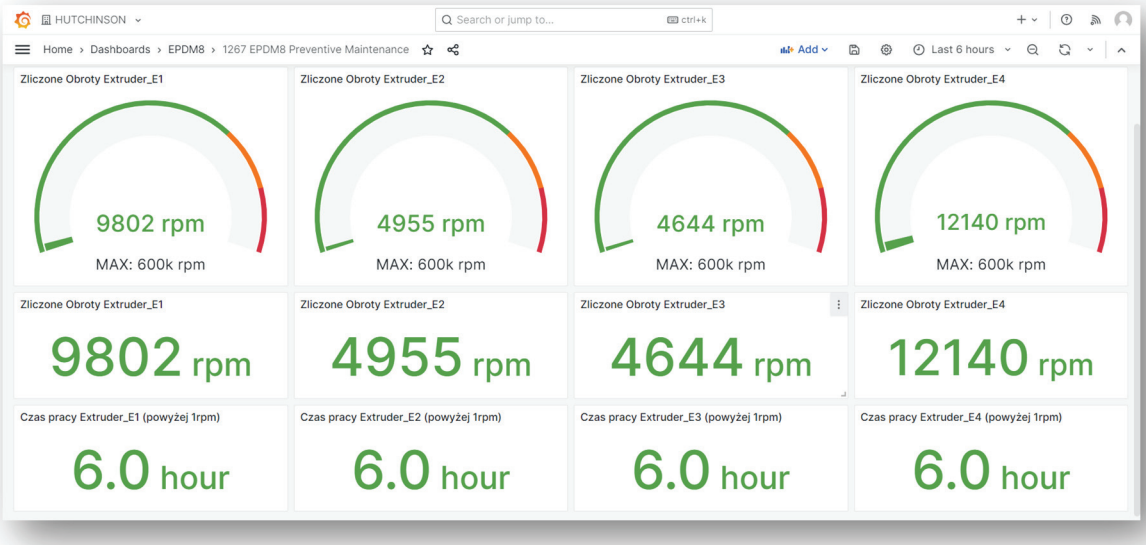


Figure 10. Final visualization

solution. Information from the user about the date of the last inspection, replacement and regeneration of the machine is useful here. Parameterization for each extruder separately may also include alarm values and rotation limits over time. Any file or table can be the basis for storing information. This solution was based on Microsoft Sharepoint lists (Figure 4). The undoubted advantage of the lists will be the ease of preparation and the ability to validate data when entered by the user. The parameterization application must ensure the identification of the parameter, and it can be used here as the previously mentioned TOPIC MQTT key, which is unique to the entire solution. Defining the time and maximum revolutions for each extruder will allow the system to be expanded with users' alarming and automatic notifications.

DISCUSSION

Any form of automation and acceleration of reporting is a very desirable element in every organization. However, the specific nature of business management always requires the presentation of an implementation plan, financing method and return on investment. In the case of this project, the investment cost, assuming that we already have server infrastructure with the ability to create virtual machines, is limited only to the purchase of professional DAD edge devices. The remaining elements are software that we can use on an open license. Preparing the solution from the IT side takes time, but it is not burdened with heavy, demanding application programming from scratch, which makes implementation easier. Experience also shows that maintaining the system does not require a staff of people with technical experience. Therefore, one of the strengths will be the low investment cost and the possibility of scaling the solution by adding new machines and parameters. Expanding the solution may allow the implementation of the assumptions of the Digital Twin concept, where key machine elements or entire production lines can be reflected in virtual models. Having information and the possibility of analysis during device operation is an increasingly appreciated functionality by experts in

the field of industrial automation. The presented reporting system model can be an excellent start for further development using artificial intelligence techniques. AI models can help detect anomalies, so the system can also be developed for real-time maintenance. Sharing a solution provides experts with a new tool, but it does not mean that it is the final solution. The success of implementing the overall solution must be verified in the future during long-term use. Based on the collected data, we can only determine the time and number of revolutions. When striving for the ideal, you also need to take into account the factor of the EPDM mixture used in production, which, due to its properties, may accelerate the wear of extruder components. By including information about the type of mixture in the process, we can take into account the wear factor, which will allow for a much more accurate determination of the expected wear of parts.

REFERENCES

1. Nițulescu, I.-V.; Korodi, A. Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions. IoT 2020, 1, 76-91. https://doi.org/10.3390/iot1010005
2. Filip, I.-D.; Iliescu, C.-M.; Pop, F. Assertive, Selective, Scalable IoT-Based Warning System. Sensors 2022, 22, 1015. https://doi.org/10.3390/s22031015
3. Manowska, A.; Wycisk, A.; Nowrot, A.; Pielot, J. The Use of the MQTT Protocol in Measurement, Monitoring and Control Systems as Part of the Implementation of Energy Management Systems. Electronics 2023, 12, 17. https://doi.org/10.3390/electronics12010017
4. Calatrava, C.G.; Fontal, Y.B.; Cucchietti, F.M.; Cuesta, C.D. NagareDB: A Resource-Efficient Document-Oriented Time-Series Database. Data 2021, 6, 91. https://doi.org/10.3390/data6080091
5. Grafana: The Open Observability Platform. Available online: https://grafana.com/
6. Available online: Mqtt.org.
7. E. Sisinni, A. Saifullah, S. Han, U. Jennehag and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," in IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-4734, Nov. 2018, doi: 10.1109/TII.2018.2852491.
8. InfluxDB: Time Series Platform for Developers. Available online: https://www.influxdata.com/
9. Brunella, V.; Aresti, V.; Romagnolli, U.; Muscato, B.; Girotto, M.; Rizzi, P.; Luda, M.P. Recycling of EPDM via Continuous Thermo-Mechanical Devulcanization with Co-Rotating Twin-Screw Extruder. Polymers 2022, 14, 4853. https://doi.org/10.3390/polym14224853
10. Available online: https://nodered.org/.