SN Computer Science (2024) 5:131
https://doi.org/10.1007/s42979-023-02442-4

ORIGINAL RESEARCH

Time Series Manufacturing Data Edge Monitoring and Visualization to Support Industrial Maintenance Teams

Ander Garcia[1] · Xabier Oregui[1] · Javier Franco[1] · Unai Arrieta[1] · Jon Ferreres[2] · Jose Andres Valencia[2]

Received: 16 April 2023 / Accepted: 19 October 2023

Abstract
Traditional manufacturing control systems such as Manufacturing Execution Systems (MES) or SCADA (Supervisory Control And Data Acquisition) were not designed for Industry 4.0 paradigm. Industry 4.0 implies that more data variables must be automatically monitored and data must be captured at a higher frequency: from one value of a few key variables to values of several variables captured at frequencies of seconds. Thus, new architectures and tools are required to merge Information Technology (IT) and Operation Technology (OT) fields and to meet Industry 4.0 requirements. This paper proposes a lightweight architecture based on micro-services and time series data requirements to connect to manufacturing process controllers, and to capture, store, monitor and visualize relevant data about the process. Moreover, a reference implementation based on Open Source tools is presented. This implementation has been validated by members of the maintenance team of a factory from Lecta, a paper manufacturer. The implementation has proven to be a new valuable tool providing further insights and customized alarms of the manufacturing process.

Keywords Cyber physical system · Industry 4.0 · Time series · Edge computing

Introduction

Pushed by the Industry 4.0 paradigm, the volume of data being captured from manufacturing lines is continuously increasing. To get a deeper insight of manufacturing processes, more data variables are being monitored and data is captured at a higher frequency: from one value of a few key variables for a whole batch, to time series of several variables captured at frequencies of seconds. Traditional Manufacturing Execution Systems (MES) were not designed for this scenario. Even traditional SCADA (Supervisory Control And Data Acquisition) systems, that control the manufacturing process and capture and visualize data at frequencies of seconds, have limited functionalities to store, analyze, and visualize data. Most SCADA systems allow operators and maintenance teams to set basic alarms to check if certain variables are out of range, but configuration and alarm rule and notification systems are usually complex to manage and offer limited functionalities. Moreover, data captured by traditional SCADA systems are complex to be exported and analyzed out of the SCADA itself.

Thus, new architectures are required to integrate Information Technology (IT) and Operations Technology (OT) fields. This implies a myriad of IT and OT technologies, standards and specifications related to Industry 4.0.

The complexity of this integration generates a knowledge barrier as these IT technologies follow a completely different philosophy from the regular tools used by OT engineers and maintenance teams. Thus, small- and medium-sized enterprises (SMEs), which generally lack multidisciplinary teams with the required IT and OT knowledge and experience, face big difficulties to capture, monitor, and visualize data from manufacturing processes.

Standard reference architectures such as RAMI 4.0 support advanced Industry 4.0 use cases, adding additional technological complexity, which does not add value for

This article is part of the topical collection "Innovative Intelligent Industrial Production and Logistics 2022" guest edited by Alexander Smirnov, Kurosh Madani, Hervé Panetto and Georg Weichhart.

✉ Ander Garcia
agarcia@vicomtech.org

[1] Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009 Donostia-San Sebastián, Spain

[2] TorrasPapel-Lecta Group, C/Elbarren, s/n, 31880 Leitza, Spain

Published online: 26 December 2023

most SMEs starting to monitor time series data from their processes.

Existing market solutions rely on external cloud servers to perform these tasks, adding a dependency on servers out of the control of manufacturing companies, which is not compatible with privacy and confidentiality requirements of several manufacturing companies.

This paper tackles this complexity by proposing a containerized micro-service-based edge architecture to monitor and visualize manufacturing processes. The architecture connects to manufacturing controllers to acquire time series data about the processes, and then stores it on a time series database to be monitored and visualized. Furthermore, a reference implementation of the architecture based on Open-Source tools is presented and validated with a simulated process. After a first successful validation of the architecture with a basic boiler simulation [5], this paper focuses on the deployment and validation of the architecture on a real industrial scenario.

This paper presents a real implementation deployed and validated at the paper factory from Lecta in Leitza. This factory specializes in carbonless, thermal, metallized, cast-coated papers, and has a production capacity of 118,000 tons per year. The implementation has been led by the maintenance team of Lecta. Currently, they have a WinCC SCADA system controlling ovens and pipes injecting hot air into the paper manufacturing line. This SCADA generates some synoptic dashboards with real-time data into a screen. However, this screen is located in a room close to the ovens but far away from the maintenance team office. Thus, as the SCADA requires someone to be in front of the screen to monitor the production process, dashboards are rarely used. Moreover, configuring alarms within the SCADA is a complex task. Current alarms were configured several years ago, when the heating system was installed. These alarms monitor key variables values that are within a predefined range, targeting potentially dangerous situations that may cause equipment to malfunction. However, daily too many alarms are fired. As nearly all of them do not harm the manufacturing line, they are ignored by the maintenance team. Finally, the SCADA has an internal database with a limited usability system to query and visualize historical data from the last month. As this system requires too much effort to analyze data, it is rarely used.

The maintenance team has validated the implementation as a new tool to enhance their daily tasks with new functionalities. The new tool has been deployed at a Linux server of their Intranet. An Ethernet link between this server and the PLC has been enabled by the IT department of Lecta to allow communications. These functionalities have concentrated the main daily use of the tool:

- Customizable dashboards accessible from any equipment of the factory
- Customizable and flexible alarms. The team has already detected some situations where several variables are involved that do not cause manufacturing problems but produce inefficiencies, such as ovens heating air directed to closed pipelines. Moreover, they want alarms to be filtered according to their severity level.
- Customizable visualizations combining variables and personalized time-ranges to analyze past states of the manufacturing process.

The structure of the paper is as follows: "Related work" reviews the related work. "Architecture" presents the proposed architecture, while "Reference implementation" introduces the reference implementation. "Implementation and validation" focuses on the implementation and validation at Lecta. Finally, "Conclusions" presents main conclusions and future work.

Related Work

The German government presented the Industry 4.0 term in 2011. The objective of the fourth industrial revolution is to work with a higher level of operational productivity and efficiency, connecting the physical to the virtual world. Industry 4.0, also known as Industrial Internet of Things (IIoT), is related to several technologies, such as Internet of Things (IoT), Industrial Automation, Cybersecurity, Intelligent Robotics, or Augmented Reality [2].

The term cyber-physical systems (CPS) was coined in the USA in 2006 and has received several definitions [4]. CPS is the merger of "cyber" as electric and electronic systems with "physical" things. The "cyber component" allows the "physical component" (such as mechanical systems) to interact with the physical world by creating a virtual copy of it. This virtual copy will include the "physical component" of the CPS (i.e., a cyber-representation) through the digitalization of data and information [2].

In general, a CPS consists of two main functional components: (1) the advanced connectivity that ensures real-time data acquisition from the physical world and information feedback from the cyber space; and (2) intelligent data management, analytics and computational capability that constructs the cyber space [6].

First attempts to integrate advanced services from Industry 4.0 on manufacturing environments were based on cloud computing. Cloud computing paradigm relies on remote servers with a storage and computing power magnitudes beyond local servers. However, cloud computing presents

four main disadvantages for manufacturing scenarios: latency, security, privacy, and cost.

Edge computing is a paradigm where data are analyzed and stored close to the devices generating and consuming them, facing previous disadvantages and making them attractive for manufacturing scenarios [1, 9].

The main objective of edge computing is to exploit computational resources of interconnected devices to increase their independence and to get data analysis and exploitation closer to where data is generated. This paradigm optimizes cloud computing paradigms moving data processing task (or part of them), to the edge of the network. This philosophy is especially relevant for manufacturing scenarios.

Edge computing devices have increasingly powerful computation functionalities. This, combines with advanced connectivity technologies such as 5G, which offers a fast, robust, and massive connectivity, is paving the way for a new type of intelligent devices and services based on Artificial Intelligence.

Recently, various attempts have been made to transform manufacturing systems into interoperable, connected, and digitalized elements. However, the main challenges of the Industry 4.0, including cybersecurity, and standardized data interchange between devices, machines and services, are still opened [7]. In [9], a review of the application of edge computing paradigm into manufacturing scenarios is provided, identifying architectures, advances, and open challenges.

Existing international reference architectures for manufacturing scenarios, such as RAMI 4.0 or IIRA, propose reference models difficult to implement [10]. Moreover, architectures proposed by other authors target a lot of complex functionalities related to the Industry 4.0 [3, 8, 11].

Thus, their implementation is time- and cost-consuming, out of the reach of small and medium manufacturing companies. The architecture proposed solves previous drawbacks to ease its deployment and targets mainly SMEs lacking IT and OT integration knowledge. The main characteristics of the proposed architecture are:

- Support time series data, to cope with the requirements of data captured from manufacturing processes.
- Edge deployment, to solve security, privacy, and latency disadvantages of cloud architectures.
- Micro-service and container-based, to decrease the knowledge barrier required to deploy the architecture.
- No-code tools reference implementation to decrease the knowledge barrier required to configure the architecture and to visualize and analyze data.

The architecture is focused on a specific case: monitor and visualize time series data from manufacturing processes. However, the architecture is flexible enough to be extended with new future services (for example to integrate Artificial Intelligence services), increase its performance, or integrate new communication and security mechanisms.

Architecture

This section presents a containerized micro-service-based edge architecture to lower the knowledge barrier required to integrate IT-OT fields and to capture, analyze, and visualize industrial data without requiring the use of cloud services. This section starts with a general description of the architecture and then presents a reference implementation based on Open Source tools.

The architecture is composed of the following components: client, message queue, writer, time series database, visualizer, and monitor (Fig. 1). The manufacturing equipment is the asset being monitored. Data from the equipment is captured from the manufacturing controller, which publishes it using standard communication specifications such as OPC-UA or MQTT.

The Open Platform Communications Unified Architecture (OPC-UA) has become the interoperability standard for the secure and reliable exchange of data in the industrial domain, easing the tasks of capturing and exporting data. In the most common OPC UA communication paradigm, the manufacturing equipment has an OPC UA server that allows to read/write variables values and to invoke custom methods. Clients connect to the server to read/write values of the variables, to call remote methods, and to subscribe to receive changes on their values.

MQTT is a robust and trustworthy protocol, with implementations with very low computation requirements and

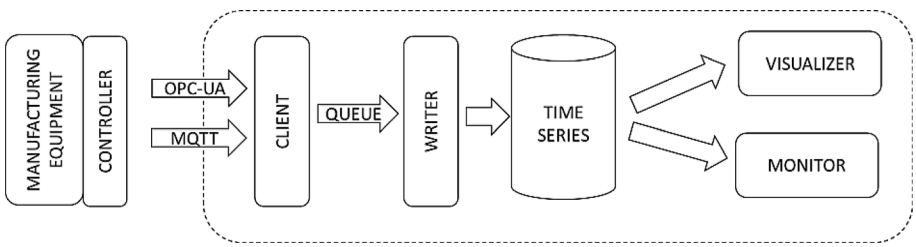


Fig. 1 General architecture

available for most of the current hardware and software platforms. MQTT is based on a queue manager (broker), where different clients send messages (publish). Each message is sent with a certain subject (topic) and may contain data (payload). Other clients can show their interest in certain topics to the que manager (subscribe). When the queue manager receives a message with some of these topics, it sends the message to the subscribed clients.

This communication paradigm based on publishing messages and subscribing to topics to receive them, has proved to be a robust, efficient, and low latency technology. Currently, it is one of the most used protocols for Internet of Things domain.

Although the proposed architecture does not impose the use of a communication protocol, authors recommend the use of OPC-UA or MQTT. Most modern Programable Logic Controllers (PLCs) already include OPC-UA or MQTT functionalities, and there are several specialized gateways on the market translating other industrial protocols to OPC-UA or MQTT.

However, if this option is not available for some manufacturing scenario, it would be always possible to develop a custom communication module inside the client to get data from the manufacturing controller.

The first element of the architecture is the client. Its main task is to integrate and translate IT and OT technologies and protocols. It connects to the manufacturing equipment using OT protocols to obtain the values of the manufacturing process and send these values using IT protocols to be stored, analyzed, and visualized. The client has to perform data cleaning and validation tasks to ensure the quality of the data, including the check of the timestamps. Moreover, when required, data has to be transformed to a proper format to be stored, for example to update numeric values to labels or Booleans, or to generate synthetic data from variables. Once data is ready, it is sent to the writer using a message queue.

The message queue decouples the client from the writer. It could be based on any technology, such as MQTT, as long as it satisfies the load requirements of each scenario.

The main task of the writer is to receive data from the message queue and to transform it into a proper format to be sent directly to the time series database to be stored.

The time series database manages data storage and retrieval operations. This database should have advanced functionalities to ease querying time series, and to aggregate data to optimize disk space utilization.

The visualizer is responsible to generate dashboards of the manufacturing processes, and to allow final users (operators, engineers…) to visualize and manually analyze data.

The last element, the monitor, is focused on the generation of alarms and notifications when data of the manufacturing process is out of its regular range, or some conditions are fulfilled.

The architecture is based on decoupled micro-services designed to be deployed as containers. The objective is (i) to ease the deployment at the edge, and (ii) to allow individual changes or upgrades of each micro-service without having to update and validate the rest of the micro-services.

Reference Implementation

This section presents a reference implementation of the previous general architecture based on Open-Source tools. Each micro-service has been designed as a Docker container, and the architecture has been orchestrated with the docker compose tool.

The client has been implemented as a Python micro-service. OPC-UA and MQTT support has been based on the FreeOpcUa library and the MQTT Paho library from the Eclipse Foundation. The client sends values of the variables to the message queue with a JSON payload with and object format. Each element of the object has three element: timestamp of the value, identifier of the equipment, and an object of data with variables name and value pairs. Thus, each message can include value for one or more variables.

The message queue has been implemented with RabbitMQ, a lightweight and widely deployed Open-Source message broker. Although it requires more computing resources than MQTT, RabbitMQ supports several messaging protocols and paradigms, and has better security and reliability features. Moreover, RabbitMQ includes internal buffers to avoid losing messages if the writer is temporarily overloaded and mechanisms to easily integrate new writer containers if several clients are sending data to the queue.

The writer has been implemented as a Python micro-service. It receives messages from the queue, and transforms data into INSERT queries for the database. This insert queries have to be formatted to fulfill the format expected by the SQL dialect of the database.

TimescaleDB has been selected as the time series database engine over other alternatives such as InfluxDB due to its advanced functionalities, SQL language compatibility, and the rich PostgreSQL-based tooling ecosystem. TimescaleDB is an Open-Source database designed to make SQL scalable for time series data. It is engineered up from PostgreSQL and packaged as a PostgreSQL extension.

Traditional relational databases, such as MySQL or SQL Server, are not suited for the storage of time series data, as their performance decrease greatly as the data volume of the time series increases. NoSQL databases, such as MongoDB, have recently include support for time series data, but the functionalities they offer to work with time series data is still not comparable to the ones offered by TimescaleDB or InfluxDB.

Finally, both the visualizer and the monitor components have been deployed based on Grafana. Grafana is a popular multi-platform Open Source analytics and interactive visualization web application. Grafana is agnostic of the underlying database and has an intuitive user interface both to customize charts and dashboards, and to generate alerts and notifications based on advanced rules and notification channels.

All the micro-services have been deployed as docker containers within the same docker network. The Web access port from Grafana has been exposed within the client host to be accessible from a Web Client. Port 5432 from TimescaleDB has also been exposed to allow the use of PostgreSQL desktop tools such as pgadmin from the host machine. Information to automatically connect micro-services and to manage data persistence of each container has been included inside the docker compose definition.

The main customization of the reference implementation to be deployed in a new scenario is related to OPC UA or MQTT, and the structure of the data and the database. For example, different OPC UA servers may send data either as an object, or as a several individual variables. Regarding MQTT, each controller may use different topic and payload definition to send data.

The design of the database is also specific of each scenario. In a general scenario, a table with these columns would be enough to store data:

- Time: to store the timestamp of the value
- Id: to store the identifier of the equipment
- Variable: To store the name of the variable
- Value: To store the value of the variable. It should be a string to allow storing different data types

However, this design may present performance drawbacks to retrieve data from the database, and to visualize and monitor it, as each value has to be parsed. Thus, it is recommended that each scenario designs its database table to store time series data.

For OPC-UA servers, a config file with the URL, and optionally the username and password, has to be updated. Moreover, a list of the identifier of each OPC UA node variable has to be filled, including the name and type of each variable. For MQTT, server connection data (URL, username and password) and the topic name have to be defined. Moreover, as MQTT payload is not standardized, code changes may be required on the client to read variable names and values from the MQTT messages.

Implementation and Validation

This section is focused on the implementation and validation of the reference implementation of the architecture in the RTO (Regenerative Thermal Oxidizer) system of the Leitza factory of Lecta. RTOs are one of the most widely accepted air pollution control technologies across industries. RTOs use a ceramic bed which is heated from a previous oxidation cycle to preheat the input gasses to partially oxidize them. The preheated gasses enter a combustion chamber that is heated by an external fuel source to reach the target oxidation temperature which is in the range between 760 and 820 °C. Thus, it is very important to recover the energy contained in the clean gasses to reduce the energy necessary for the treatment. To recover this energy, the thermoreactor uses ceramic beds that act as heat accumulators, capturing the heat of the exhaust gasses to transfer it, later, to the gasses to be treated. The RTO consists of three towers filled with ceramic elements, communicated at the top by the oxidation chamber.

The RTO SCADA system monitors 123 variables from a Siemens 1200 PLC, which controls the process. These variables include set and real values of temperatures, pressures, air flows and state of air pumps (Fig. 2).

The reference implementation (Fig. 3) has been customized to communicate with the Siemens 1200 PLC. This PLC does not provide OPC UA or MQTT functionalities, it communicates using the custom Siemens protocol S7. Thus, instead of including a gateway or a translation module from S7 to OPC UA or MQTT, a new S7 client has been developed in Python. This client is based on the Python-snap7 library, a wrapper to Snap7, a multi-platform Ethernet communication suite for interfacing natively with Siemens S7 PLCs. This client loads a CSV file with the name, type, maximum allowed value (optional), minimum allowed value (optional), and memory position of each of the 123 variables and periodically (2 s) reads their value. Instead of generating 123 queries to the PLC, variables are grouped according to the data block of the PLC they belong to. This way only 12 queries are sent per 2 s.

Once data of each variable has been read, the client send it to a RabbitMQ queue named “rto”. The next code shows an example partial payload of the message sent by the client:

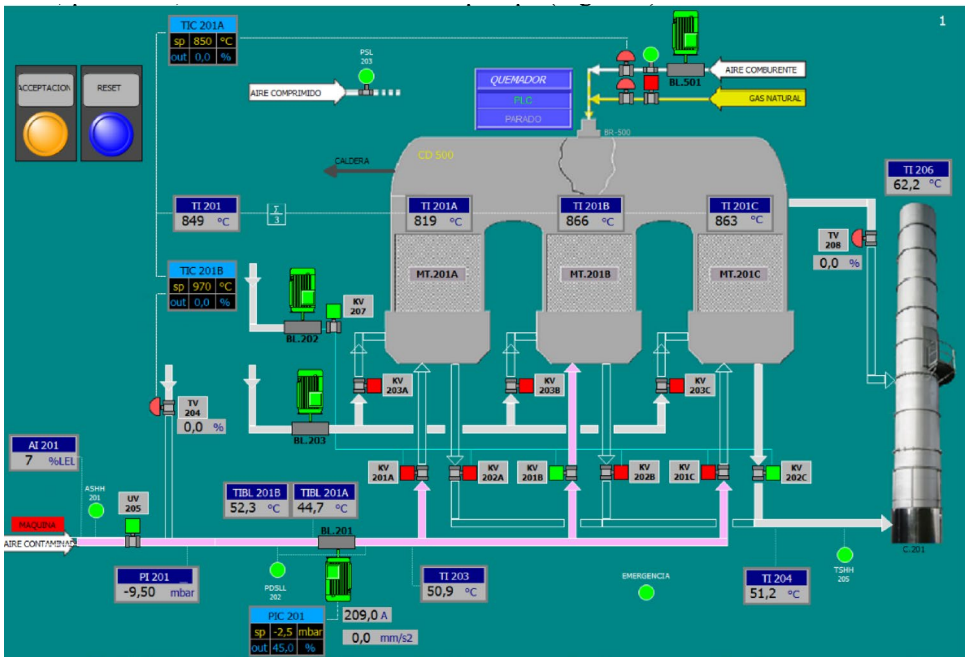


Fig. 2 Dashboard from the RTO SCADA

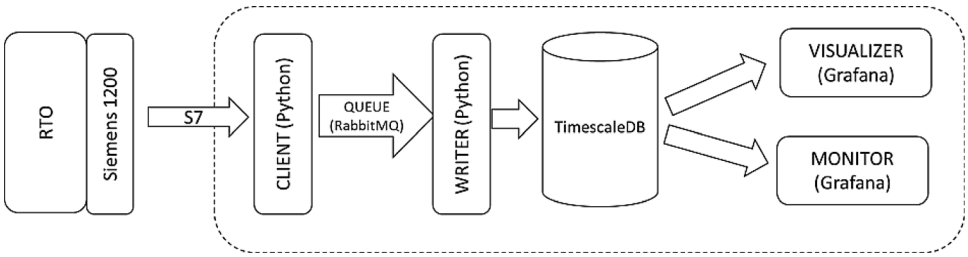


Fig. 3 Deployed implementation

the client:

```
{
   "data": {
      "depuration": 31,
“plant_state”:4,
“Kv201a_opt”: 5330,
“kv203c_opt”:878,
....
}
},

   },
   "time": "2023-03-12T10:15:18.784Z",
   "id": "rto"
}
```

The writer receives these updates and sends them to the TimescaleDB database. Based on the previous CSV, a table has been created for the RTO with the following columns:

- Time: to store the timestamp of the value
- Id: to store the identifier of the RTO. As there is only one RTO, its value is always the same.
- One column for each of the 123 variables of the PLC

The writer receives each message and generates the following SQL query to insert data. The database receives the query and stores data on the table (Fig. 4).

	time timestamp without time zone	depuration real	plant_state real	kv201_a real	kv201a_opt real	kv201_b real	kv201b_opt real
1	2023-04-12 14:33:29.906402	31	4	1	5330	0	4798
2	2023-04-12 14:33:33.573325	31	4	1	5330	0	4798
3	2023-04-12 14:33:36.030313	31	4	1	5330	0	4798
4	2023-04-12 14:33:38.545519	31	4	1	5330	0	4798
5	2023-04-12 14:33:40.877809	31	4	1	5330	0	4798

Fig. 4 Screenshot of a partial select query of the data

Fig. 5 Dashboard for pew272 variable

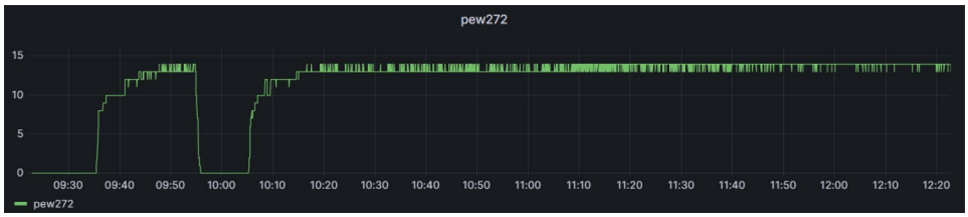
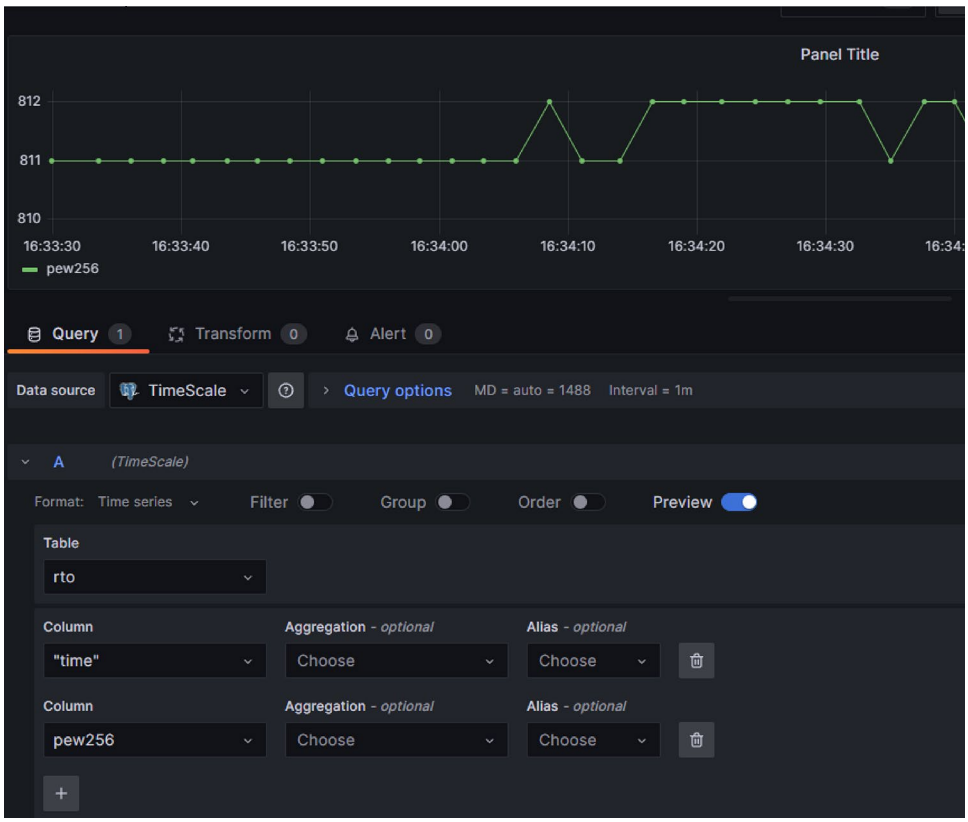


Fig. 6 Query builder from Grafana



INSERT INTO rto("time", "depuration", "plant_state", "kv201_a", "kv201a_opt",....,"id") VALUES ('2023-04-12T12:18:03.779Z', 30, 4, 1,5325,...,'rto');

The visualizer has been deployed as the latest version of Grafana. After configuring the data source to connect to the Timescaledb database, the previous CSV has been used to automatically generate one dashboard for each variable (Fig. 5).

Definition of each graph is easily customized using the available query builder (Fig. 6). Using the GUI a general SQL query is generated, and it is also possible to insert manual SQL queries to directly integrate advanced functions from TimescaleDB such as time buckets. Time buckets allow to get uniformly distributed data points within a range, for example one value with the average of the values from the database every 10 min.

The maintenance team has been trained to easily generate new dashboards that visualize and combine several variables without requiring support from IT staff. Moreover, as each dashboard is linked to a unique URL that can be shared and is accessible from any browser within the factory, the maintenance team can generate custom visualization for certain users: operators, managers…

The notifier has also been integrated with Grafana. Based on the previous CSV, one alarm has been configured to fire for each of the variables with a maximum and minimum value. These alarms check the value of the variable is out of the predefined range. The name of each alarm is the name of the variable, and all of them have been assigned the same priority level. If some alarm is fired for a certain time, and email is sent to the maintenance team. The maintenance team can customize the priority level, contact points and notification policies of each alarm.

Moreover, three new alarms have been configured to monitor certain situations that led to breakdowns or efficiency losses.

- Emergency variable. The first alarm is focused on the emergency variable. This is the most basic alarm, as it only checks the value of this variable is below 0.1
- Air losses. These alarms focus on situations where air pumps are closed but airflow is detected. This airflow is caused by a pump malfunction that leads to hot air losses, and thus is wasting energy. These alarms combine the value of the detected air flow and the state of the air pumps. If there is a considerable air flow and the air pumps are supposed to be closed, an alarm is fired to notify an efficiency loss.
- Boiler fill time. From time to time, the boiler must be filled to avoid breakdowns. An excess of the filling time is a sign that a maintenance operation is required to avoid

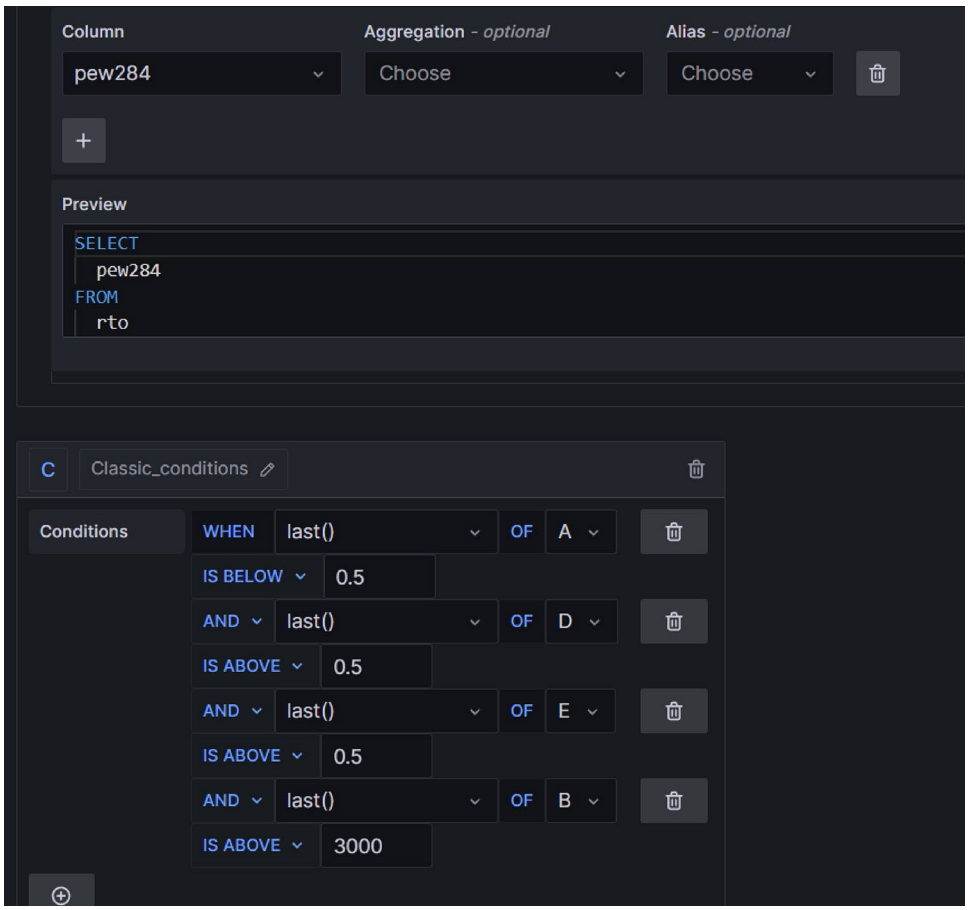


Fig. 7 Example of a complex rule generation

further problems. This alarm analyzes the filling time after the minimum level has been reached and fires it is exceeded an stablished threshold.

The alarms are easily configured using the Grafana GUI (Fig. 7). For each alert rule, after selecting the data source and the related table and column, several conditions can be applied to decide whether an alert should be raised.

Grafana has a powerful alert customization and notification mechanism able to suit most of the regular requirements to monitor manufacturing equipment. Once rules have been defined, labels can be attached to them to ease their management. Then, a notification policy is applied where several filters regarding time, labels, severities… allow to decide whether the alert has to be redirected to any of the available notification channels. There are several notification channels (email, slack, PagerDuty…) available, and custom ones can also be defined.

The whole system has been defined as docker containers orchestrated within a docker compose file. This docker compose file can be used as a template to be deployed in new scenarios, after updating the points mentioned in the previous section. The deployment is running on a server of the intranet from Lecta running a Ubuntu 22.04.1 LTS on a 8 GB RAM and 4 CPU virtual machine inside an Intel(R) Xeon(R) CPU E5-2440 0 @ 2.40 GHz server.

The maintenance team from Lecta has successfully validated the system to perform three main tasks:

- Generate customized dashboards
- Generate customized and flexible alarms.
- Generate customized visualizations

Furthermore, the system has confirmed with data that efficiency loss situations detected by the experience of the operators were actually happening. Thus, these situations can be detected and measures to avoid them can be planned.

Conclusions

Industry 4.0 requires data to get insights of the manufacturing processes. Thus, requirements to capture more data variables and at a higher frequency arise: from one value of a few key variables for a whole batch, to time series of several variables captured at frequencies of seconds. Traditional Manufacturing Execution Systems (MES) were not designed for this scenario composed by a high volume of time series data of manufacturing processes. Even traditional SCADA (Supervisory Control And Data Acquisition) systems, that control the manufacturing process and capture and visualize data at frequencies of seconds, have limited functionalities to store, analyze, and visualize data.

Thus, new architectures are required to integrate Information Technology (IT) and Operations Technology (OT) fields. This implies a myriad of IT and OT technologies, standards and specifications related to Industry 4.0, with a high complexity level. SMEs are not ready to cope with this complexity level.

This paper tackles this complexity by proposing a containerized micro-service-based edge architecture to monitor and visualize manufacturing processes. The architecture connects to manufacturing controllers to acquire time series data about the processes, and then store it on a time series database to be monitored and visualized. A reference implementation of the architecture based on Open-Source tools has been presented. Moreover, the implementation has been deployed and validated at the Leitza paper factory of Lecta to monitor the RTO system.

The architecture is based on decoupled containers to be easily deployed at the edge. It has four main elements. The client is the component integrating OT and IT domains: it connects to the manufacturing equipment using OT technologies to obtain the values of the manufacturing process. Once data is ready, it is sent to the writer using a message queue, already within the IT domain.

The main task of the writer is to receive data from the message queue and to transform it into a proper format to be sent directly to the time series database to be stored. The time series database manages data storage and retrieval operations.

The visualizer is responsible to generate dashboards of the manufacturing processes, and to allow final users (operators, engineers…) to visualize and manually analyze data. The last element, the monitor, is focused on the generation of alarms and notifications when data of the manufacturing process is out of its regular range, or some conditions are fulfilled.

A reference implementation based on the following Open Source components has also been provided:

- Custom Python scripts for the client and the writer
- RabbitMQ message queue to connect the client and the writer
- TimescaleDB to store time series data
- Grafana to deploy and customize the visualizer and the monitor

This implementation has been validated monitoring the RTO system of the factory. Data from the RTO has been captured from the Siemens 1200 PLC controlling the process. Communication with the PLC is based on Siemens S7

protocol. Using S7, 12 queries per 2 s are generated to read the value of 123 variables of the process. Information of the variables has been defined at a CSV. This CSV has been used to automate the generation of the client reading the values of the variables, the generation of the table of the database, and basic Grafana dashboards and alarms of the variables.

The proposed architecture greatly decreases the technological barrier required to monitor and visualize data from manufacturing processes. Moreover, as data is already properly stored at the database, it serves as a foundation for future services, for example integrating Artificial Intelligence algorithms to provide predictive maintenance functionalities.

The maintenance team from Lecta has been provided with new data-based tools that support them to fire customized alarms, detect anomalies, and confirm some problems were actually happening. Furthermore, the new tool is a solid starting foundation to deploy new data and AI-based services on the future.

Future work starts with further validation at Lecta for a relevant period of time to test the resilience and scalability of the implementation. One of the main purposes of this validation will be to estimate the disk space required by the system. With the current query period of 2 s, the system generates more than 40,000 rows per day. However, the time granularity can be decreased to generate views with less frequently updated data (for example one data per minute for not recent data) to save disk space. This validation will help to set adequate aggregation and retention policies.

Another open work line focuses on quantifying efficiently losses caused by detected situations, to measure their economic impact and to decide if related equipment should be upgraded.

One last point to further decrease the technological barrier consists of the integration of no-code tools, such as node-red. Node-red is a popular graphical tool where non-expert users interact with simple blocks to customize the functionalities of a system using an interactive interface.

Funding This work has been partially founded by the Basque Government (SPRI) through the following Elkartek project: KK-2021/00111 ERTZEAN.

Declarations

Conflict of interest All authors declare they did not have conflict of interests.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Alam M, Rufino J, Ferreira J, Ahmed SH, Shah N, Chen Y. Orchestration of microservices for IoT using docker and edge computing. IEEE Commun Mag. 2018;56(9):118–23. https://doi.org/10.1109/MCOM.2018.1701233.
2. Alcácer V, Cruz-Machado V. Scanning the Industry 4.0: a literature review on technologies for manufacturing systems. Eng Sci Technol Int J. 2019;22(3):899–919. https://doi.org/10.1016/j.jestch.2019.01.006.
3. Azarmipour M, Elfaham H, Gries C, Kleinert T, Epple U (2020) A service-based architecture for the interaction of control and MES systems in industry 4.0 environment. In: IEEE international conference on industrial informatics (INDIN), 2020-July, pp. 217–222. https://doi.org/10.1109/INDIN45582.2020.9442083.
4. Fei X, Shah N, Verba N, Chao KM, Sanchez-Anguix V, Lewandowski J, James A, Usman Z. CPS data streams analytics based on machine learning for cloud and fog computing: a survey. Future Gener Comput Syst. 2019;90:435–50. https://doi.org/10.1016/j.future.2018.06.042.
5. Garcia A, Oregui X, Franco J, Arrieta U (2022) Edge containerized architecture for manufacturing process time series data monitoring and visualization. In: 3rd International conference on innovative intelligent industrial production and logistics (IN4PL 2022), pp. 145–152
6. Lee J, Bagheri B, Kao HA. A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. Manuf Lett. 2015;3:18–23. https://doi.org/10.1016/j.mfglet.2014.12.001.
7. Lu Y. Industry 4.0: A survey on technologies, applications and open research issues. J Ind Inf Integr. 2017;6:1–10. https://doi.org/10.1016/j.jii.2017.04.005.
8. Omar A, Imen B, M'Hammed S, Bouziane B, David B (2019) Deployment of fog computing platform for cyber physical production system based on docker technology. In: Proceedings—2019 3rd international conference on applied automation and industrial diagnostics, ICAAID 2019, 1(September), pp. 1–6. https://doi.org/10.1109/ICAAID.2019.8934949.
9. Qiu T, Chi J, Zhou X, Ning Z, Atiquzzaman M, Wu DO. Edge computing in industrial internet of things: architecture, advances and challenges. IEEE Commun Surv Tutor. 2020;22(4):2462–88. https://doi.org/10.1109/COMST.2020.3009103.
10. Szántó N, Pedone G, Monek G, Háy B, Jósvai J. Transformation of traditional assembly lines into interoperable CPPS for MES: an OPC UA enabled scenario. Procedia Manuf. 2021;54:118–23. https://doi.org/10.1016/j.promfg.2021.07.019.
11. Yang C, Lan S, Shen W, Wang L, Huang GQ. Software-defined cloud manufacturing with edge computing for Industry 4.0. 2020 international wireless communications and mobile computing. IWCMC. 2020;2020:1618–23. https://doi.org/10.1109/IWCMC48107.2020.9148467.