Contents

1 Introduction **1**
- 1.1 Data Analysis . . . 1
  - 1.1.1 Procedure for analysing data . . . 2
  - 1.1.2 What is Data? . . . 4
  - 1.1.3 Connection to the Scientific Method . . . 5
- 1.2 Operational environment: Maintenance . . . 6
  - 1.2.1 Preventive Maintenance . . . 7
  - 1.2.2 Condition-based maintenance & monitoring . . . 9
  - 1.2.3 A possible solution . . . 11

2 The hosting company: Zensor **13**
- 2.1 Maintenance partner . . . 13
- 2.2 Zensor Approach . . . 14
- 2.3 Workflow . . . 17

3 Tools for data analysis **21**
- 3.1 Pandas . . . 21
  - 3.1.1 Series and DataFrame . . . 21
  - 3.1.2 Core Features . . . 22
- 3.2 InfluxDB . . . 25
  - 3.2.1 TSDB: time series database . . . 26
  - 3.2.2 Influx solution . . . 27
- 3.3 Grafana . . . 29
  - 3.3.1 Dashboard: what is it? . . . 29
  - 3.3.2 Key strengths . . . 30

4 Use cases of interest **33**
- 4.1 Analyse blade grinder vibration . . . 33
  - 4.1.1 Initial Hypothesis . . . 33
  - 4.1.2 Goal(s), purpose & critical factors . . . 34
  - 4.1.3 Project description by phases . . . 35
  - 4.1.4 Conclusion . . . 43
- 4.2 Monitor electricity consumption . . . 44
  - 4.2.1 Initial Hypothesis . . . 44

4.2.2 Goal(s), purpose & critical factors 45
4.2.3 Project description by phases 46
4.2.4 Conclusion 49

A Acronyms **51**

List of Tables **53**

List of Figures **54**

Bibliography **55**

Chapter 1

Introduction

In this opening chapter, we take a closer look at two major research areas data-analysis and maintenance, and try to highlight where they intersect. This chapter is divided into two parts:

1. First, to set the scene, we will give a definition of data analysis and briefly review the main steps involved in the process.

2. Second, we will talk about maintenance, how it has changed over time, and what benefits data analysis techniques can bring to the industry.

1.1 Data Analysis

Data analysis is the act of analysing, cleansing, manipulating, and modelling data in order to identify usable information, generate conclusions, and help decision-making processes [1]. In today's corporate world, data analysis plays an important part in making decisions more scientific and assisting firms in operating more efficiently.

Data analysis has several dimensions and approaches, including a wide range of techniques known by various names and applied in a variety of business, science, and social science sectors [2]. Let's give some examples. *Data mining* (DM) is a type of data analysis technique that focuses on statistical modelling and knowledge discovery for predictive rather than purely descriptive purposes, whereas *Business Intelligence* (BI) is a type of data analysis that focuses on aggregation and is primarily concerned with business information. Furthermore, in statistical applications, Data analysis can be separated into descriptive statistics, *Exploratory data analysis* (EDA), and *Confirmatory data analysis* (CDA) [3]. EDA is concerned with finding new features in data, whereas CDA is concerned with validating or refuting current assumptions [4]. Finally, predictive analytics focuses on the application of statistical models for predictive forecasting or classification, while text analytics, applies statistical, linguistic, and structural techniques to

extract and classify information from textual sources, a species of unstructured data. All the aforementioned are examples of data analysis [5].

1.1.1 Procedure for analysing data

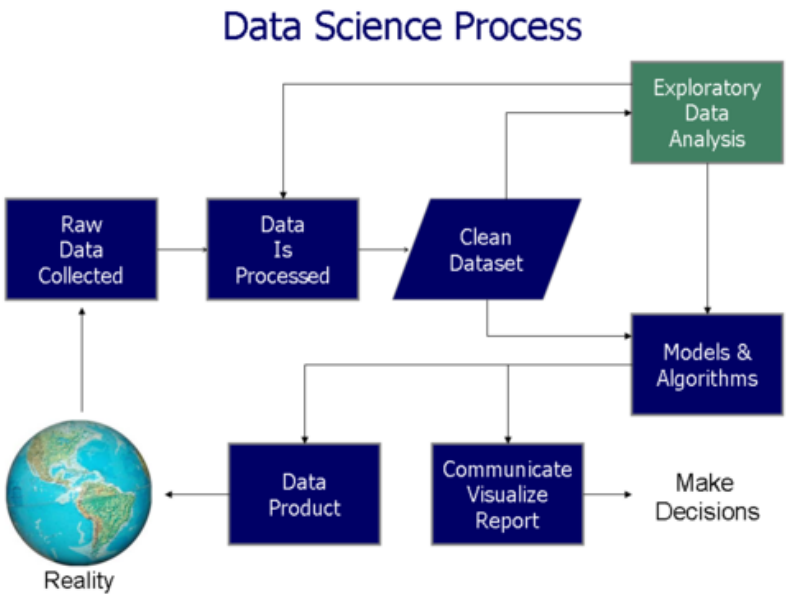


Figure 1.1: Data science process flowchart (Source: [3])

The term *"analysis"* refers to the process of breaking down a whole into its constituent parts for closer evaluation. Data analysis is the act of getting raw data and then transforming it into information that users can utilize to make decisions [1]. Data is gathered and processed in order to answer questions, test hypotheses, or refute theories. Statistician Tukey, defined data analysis in 1961 [6], as following:

> "Procedures for analysing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analysing data."

There are various distinct phases that can be identified as show in Figure 1.1, these are iterative in the sense that input from later phases may lead to further effort in earlier ones [3]. Next, we are going to present them one by one, in a slightly more detailed manner, stressing that similar stages can be found in the *Cross-industry standard process* (CRISP) framework, which is used in Data mining.

Data Collection

Data is collected from a broad variety of sources, like sensors in the environment, including traffic cameras, satellites, recording devices, etc. and it may also be obtained through interviews, downloads from online sources, or reading documentation [3].

Data Processing

Data, when initially obtained, must be processed or organized for analysis. For instance, these may involve placing data into rows and columns in a table format (known as structured data) for further analysis, often through the use of spreadsheet or statistical software [3].

Data Cleaning & Cleansing

After it has been processed and structured, data may be missing, duplicated, or contain errors. Data cleaning will be essential as a result of challenges with the way data is entered and stored. The process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database is, indeed, known as data cleansing (or *cleaning*), and it entails identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data [7]. The actual data cleaning workflow may include removing typographical errors and/or validating and correcting values against a known list of entities.

Exploratory data analysis

Once the datasets are cleaned, they can then be analysed. EDA is the first step toward building a model, since it is a critical part of the data science process, and also represents a philosophy developed by Tukey in contrast to CDA, which concerns itself with modelling and hypotheses [6]. Indeed, in EDA, there is no hypothesis and there is no model. The "exploratory" aspect means that your understanding of the problem you are solving, or might solve, is changing as you go.

The process of data exploration may result in additional data cleaning or additional requests for data; thus, the initialization of the iterative phases mentioned in the lead paragraph of this section. Data visualization is also a technique used, in which the analyst is able to examine the data in a graphical format in order to obtain additional insights, regarding the messages within the data [3].

Modelling and algorithms

Mathematical formulas or models (known as algorithms) can be applied to data to identify relationships between variables, such as correlation or

causation [2]. In general, models can be created to evaluate a specific variable based on other variables in the dataset, with some residual error depending on the accuracy of the implemented model (e.g., $Data = Model + Error$) [8].

Data product

A data product is a computer application that takes data inputs and generates outputs, feeding them back into the environment. As such it may be based on a model or algorithm. For instance, an application that analyses data about customer purchase history, and uses the results to recommend other purchases the customer might enjoy [3].

Data visualization & Communication

Once data is analysed, it may be reported in many formats to the users of the analysis to support their requirements [4]. The users may have feedback, which results in additional analysis. As such, much of the analytical cycle is iterative, as stated before. A company dashboard, for instance, that visualizes some *Key Performance Indicators* (KPIs), is both a data product and a decision support system, with a nice user interface that allows to access multiple visualization 3.3.

1.1.2 What is Data?

Variable	Type(s)	Description	Examples
Categorical	Nominal	Named categories with no implied order	Blood groups, breed, gender, neuter status
	Ordinal	Ordered categories where the differences between categories are not necessarily equal	Scoring systems, cancer staging, onset of disease (peracute, acute, chronic)
Continuous	Interval	Equal distances between values, but the zero point is arbitrary	IQ, ordinal data with equal-appearing categories
	Ratio	Above as for interval and a meaningful zero; data usually obtained by measurement	Weight, age, temperature, blood pressure

Table 1.1: NOIR system of classification of types of data (Source: [4])

We discussed at length about data in the previous subsection; certainly it is important that one have the ability to analyse and investigate data, but understanding data is an important prerequisite for being able to use it properly, and perhaps the single most important element. Fortunately the NOIR system, commonly used, can help us. It defines the type of data as nominal, ordinal, interval or ratio as show in table 1.1.

So we can confirm that valuable data is no longer only a collection of numbers and classified variables and a strong data scientist needs to be versatile and comfortable dealing with a variety of types of data, including:

- Traditional: numerical, categorical, or binary.
- Text: emails, tweets, New York Times articles.
- Records: user-level data, timestamped event data, log files.
- Complex: Geo-based location data (GIS), Network & Images.
- Sensor data, my use-case (see Chapter 4).

1.1.3 Connection to the Scientific Method

In both the data science process and the scientific method, not every problem requires one to go through all the steps, but almost all problems can be solved with *some* combination of previously mentioned stages [3]. In fact, We can think of the data science process as an extension of or variation of the scientific method:

- Ask a question.
- Do background research.
- Construct a hypothesis.
- Test your hypothesis by doing an experiment.
- Analyse your data and draw a conclusion.
- Communicate your results.

As an example, if your end goal is a **data visualization** (which itself could be thought of as a data product), it's possible you might not do any machine learning or statistical modelling, but you would want to get all the way to a clean dataset, do some Exploratory data analysis, and then create the visualization. This has happened to me many times during my internship, as we will see in Chapter 4.

Data-driven context There are numerous areas in which data analysis shines, but in this text we will focus on one specific area: maintenance. For complex systems such as aeroplanes, railways, power plants, is a big issue (and challenge) as it ensures the system reliability and safety during their life cycles.

But what does the term maintenance mean and which and how many types exist?

1.2 Operational environment: Maintenance

In industrial, business, and domestic settings, maintenance entails functioning checks, maintaining, repairing, or replacing necessary devices, equipment, machinery, building structures, and supporting utilities. This has evolved over time to encompass a variety of terms that indicate various cost-effective techniques for keeping equipment operating; these actions might occur before or after a failure [9].

Types The marine and air transportation, offshore structures, industrial plant and facility management industries depend on *Maintenance, repair and overhaul* (MRO) including scheduled or preventive paint maintenance programmes to maintain and restore coatings applied to steel in environments subject to attack from erosion, corrosion and environmental pollution [10].

Architectural conservation employs MRO to preserve, rehabilitate, restore, or reconstruct historical structures with stone, brick, glass, metal, and wood which match the original constituent materials where possible, or with suitable polymer technologies when not. The basic types of maintenance falling under MRO include:

- **Corrective maintenance**, where equipment is repaired or replaced after wear, malfunction or break down.
- **Preventive maintenance**, where equipment is checked and serviced in a planned manner (in scheduled points in time or continuously).
- **Reinforcement**, where equipment is reinforced and hardened to prevent failure.

These are huge topics, and we are going to focus mostly on corrective and preventive maintenance.

Corrective maintenance Is a type of reactive maintenance that is performed on equipment after it has broken down or malfunctioned, sometimes referred to as "fighting fires". Not only can wear equipment damage other parts and cause multiple damages, but it can also result in significant repair and replacement costs as well as lost revenue due to downtime during

Figure 1.2: *USS Ronald Reagan (CVN 76) Undergoes Preventive Maintenance* (Source: [11])

overhaul. Traditional procedures like welding and metal flame spraying, as well as designed solutions with thermoset polymeric materials, are used to rebuild and resurface equipment and infrastructure damaged by erosion and corrosion as part of corrective or preventive maintenance programs.

1.2.1 Preventive Maintenance

Preventive maintenance (PM) is, according to Decourcy Hinds [12]:

> *. . . a routine for periodically inspecting with the goal of noticing small problems and fixing them before major ones develop.*
>
> *Ideally, nothing breaks down!*

The main objectives of preventive maintenance are as follows:

1. **Enhance** capital equipment productive life.
2. **Reduce** critical equipment breakdown.
3. **Minimize** production loss due to equipment failures.

Many people, me included, confuse the phrases *preventive, predictive, and prescriptive* maintenance, and while they are distinct, the latter two might be considered kinds of preventive maintenance. Preventative maintenance in all forms aids manufacturers in transitioning from a repair-and-replace to a preventive maintenance approach [13]. Let's look at the different types of preventative maintenance.

- ***Planned preventive maintenance*** **(PPM)**, more commonly referred to planned maintenance or scheduled maintenance, is any variety of scheduled maintenance to an object or item of equipment. Specifically, planned maintenance is a scheduled service visit carried out by a competent and suitable agent, to ensure that an item of equipment is operating correctly and to therefore avoid any unscheduled breakdown and downtime. It can be further split in two subcategories:

  - **Calendar-based maintenance** is performed on equipment according to a calendar timetable. In other words, a maintenance activity is triggered by the passage of time. Calendar-based maintenance includes things like: cleaning your air conditioner and replacing the air filter in your heating, ventilation, and air conditioning equipment every three months.

    When a scheduled task is due, *Computerized Maintenance Management Systems* (CMMS) are frequently used to keep schedules straight and issue recurring work orders.

  - **Usage-based maintenance** uses triggers based on how much each piece of equipment is used. Maintenance managers can create a preventative maintenance schedule based on predefined parameters by tracking usage using equipment monitors and the operating hours of each piece of machinery.

    For example, when machine X reaches a certain number of hours of operation, this creates a trigger to schedule a booking for a service technician to perform on-demand maintenance.

- ***Predictive maintenance*** **(PdM)** are intended to assist in determining the state of in-service equipment, so that maintenance can be scheduled. Because actions are performed only when warranted, this strategy promises cost reductions over routine or time-based preventive maintenance. As a result, it is viewed as condition-based maintenance which is carried out in accordance by estimations of an item's degradation status. Predictive maintenance's key benefit is that it allows for easy scheduling of corrective maintenance and prevents unexpected equipment breakdowns [14].

  It is particularly useful when coupled with CMMS software; Logging work requirements data allows managers to review data and notice failure patterns over time. This information allows predicting when outages will occur based on historical data and plan maintenance tasks to avoid them [13].

- ***Condition-based maintenance*** **(CBM)** to put it succinctly, is *maintenance when it is needed*. Although being chronologically much older, it is considered one sector or practice within the broader and younger

predictive maintenance field, where new *Artificial Intelligence* (AI) technology and *Internet Of Things* (IoT) connectivity abilities are put to use, to help schedule preventive maintenance tasks. CBM is performed after one or more indicators show that equipment is going to fail or that equipment performance is deteriorating; this concept is applicable to both mission-critical systems that incorporate active redundancy and fault reporting, and non-mission-critical systems that have a more limited budget; let's explore this in a little more detail.

1.2.2 Condition-based maintenance & monitoring

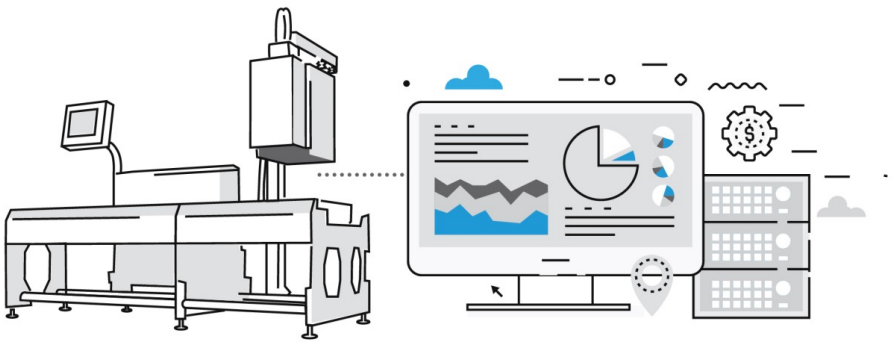


Figure 1.3: Condition-based maintenance schema (Source: [15])

CBM was developed to ensure that the proper equipment was maintained at the right time. It is focused on prioritizing and optimizing maintenance resources using real-time data, through *Condition monitoring* (CM), which is the process of observing the system's state; how? Through monitoring parameter(s) of condition in machinery (vibration, voltage, temperature etc.), in order to identify a significant change which is indicative of a developing fault [16]. The equipment's health will be determined by this system, and maintenance will be performed only when it is genuinely required.

Recent technological advancements have enabled widespread equipment instrumentation, and, when combined with improved tools for analysing condition data, maintenance personnel is more than ever able to determine when it is appropriate to undertake maintenance on a piece of equipment. CBM, in essence, should allow maintenance professionals to focus on only the necessary tasks, reducing spare parts costs, system downtime, and maintenance time. Furthermore, using *Machine Learning* (ML) based scheduled

maintenance could not only predict a possible failure, but also attempts to make result-oriented maintenance recommendations based on that machine's analysis.

Challenges Despite its usefulness, there are several challenges to the use of Condition-based maintenance.

1. First and most important of all, the initial cost can be high since it requires improved instrumentation of the equipment. Often the cost of sufficient instruments can be quite large, especially on equipment that is already installed, even though wireless systems have reduced the initial cost. Therefore, it is important for the installer to decide the importance of the investment before adding CBM to the equipment.

   For instance, a result of this cost is that the first generation of CBM in the oil and gas industry has only focused on vibration in heavy rotating equipment [17].

2. Secondly, introducing this process will invoke a major change in how maintenance is performed, and potentially to the whole maintenance organization in a company. As we well know organizational changes are in general difficult.

3. Lastly, the technical side of it is **not** simple! Even if some types of equipment can easily be observed by measuring simple values such as vibration (displacement, velocity or acceleration), temperature or pressure, it is not trivial to turn this measured data into actionable knowledge about the health of the equipment.

Advantages and disadvantages Condition-based maintenance has some advantages and disadvantages over Planned preventive maintenance:

Pros	Cons
Improved system reliability	High installation costs, for minor equipment items often more than the value of the equipment
Decreased maintenance costs	Unpredictable maintenance periods cause costs to be divided unequally
Decreased number of maintenance operations causes a reduction of human error influences	Increased number of parts (the CBM installation itself) that need maintenance and checking

1.2.3 A possible solution

Today, most often, technical data sheets coupled with the knowledge of a number of unique experienced individuals are used to determine when the asset requires maintenance. Product quality only becomes an issue when customers start complaining and repairs are done when it is already far too late, All of these puts tremendous strain on the people responsible for the asset, while it could be avoided Unexpected shutdowns are costly and very demanding for the workforce involved; a possible solution would be making assets smart in order to increase the availability. The best way to validate the actual health is by having a continuous look at a broad data set and adding a specific multi-aspect monitoring setup consisting of different sensor types that follow the behaviour of the asset's general state-of-health [18].

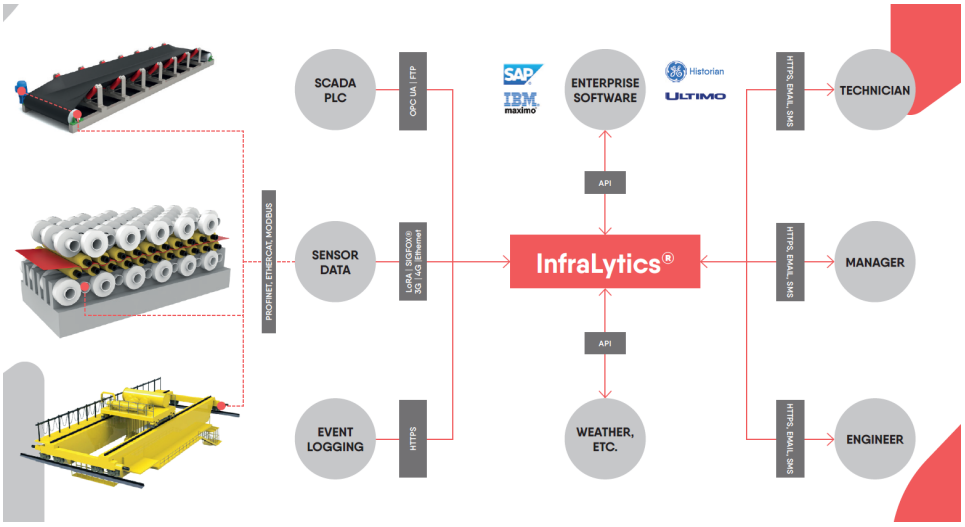


Figure 1.4: *Service as a Service* Workflow (Source: [18])

Infrastructure Analytics **(Infralytics©)** Zensor platform 1.4 is designed to make PdM as easy as possible. It graphically shows you the current state of your asset, it alarms you when a certain component is about to break down and the collection and aggregation of all kinds of sensor and operational data makes it possible to go beyond the symptom level of a mechanical failure and detect the underlying reason for the breakdown. Relying on Infralytics© will not only lower your maintenance costs significantly, it will also have a serious impact on your plant's efficiency by increasing uptime and lowering the occurrence of unexpected mechanical breakdowns [19].

Chapter 2

The hosting company: Zensor

As clearly stated in the chapter title my Erasmus$^+$ Traineeship was hosted by Zensor. This chapter, dived in three sections, offers an overview of:

1. what benefits Zensor can bring to the industrial sector and his players;
2. how is Zensor's service organized and what are its objectives;
3. lastly, what is the workflow of a typical project.

2.1 Maintenance partner

Zensor [18] provides full, integrated and intelligent monitoring solutions for the industrial production, renewable energy and infrastructure sectors. This allows to Zensor's customers to convert the potential of the world of IoT and Industry 4.0 into their reality.

The company enables digitalization, but with an interface oriented towards the real human: an expert solution without the need for internal experts, covering one or more of the elements shown in Table 2.1.

It provides not only monitoring devices or data analysis, but it offers a full, standardized and standalone end-to-end product that leverages the value contained in a subset of the following aspects:

* Operational efficiency.
* Energy efficiency.
* Predictive maintenance.
* Ageing and degradation.
* Safety.

Vision Technological advance can only bring real value to society when the user-facing component is driven by simplicity and clarity for the end-user. Zensor sees this as the fastest and most certain route to a world where man-made structures affect the sustainability of our planet in the least possible way:

Asset	Industries	Infrastructure
Rolling cranes	Metal Production	Offshore wind
Grinders and crushers	Mining and Materials	Rail
Flattener rollers	Food Production	Civil Infrastructure
Rolling mills	Glass Production	Energy
Conveyor belts	Discrete Manufacturing	
Tunnels	Textiles	
Chain transporters		
Sieves		
Bridges		

Table 2.1: Assets, Industries, and Infrastructure for which Zensor has specific products

- they are intrinsically safe;
- their useful life is optimized to the maximum;
- their impact on environment and society is quantified and communicated.

Mission Zensor's mission is to translate technological innovations in monitoring and analysis into easy-to-understand, tangible and relevant information that we share in the way tailored for either production managers, management, or maintenance professionals ...

As such, we are the knowledgeable and easy-to-reach companion for owners and operators in making their assets increasingly safe, efficient and sustainable. If up to us, till eternity.

2.2 Zensor Approach

A Zensor's standard offering can consists, if required, in several aspects, ordered logically here:

1. `Hardware`: sensors and acquisition units.
2. `Installation and Commissioning`: engineering and CAD.
3. `Data Management:` data transfer & storage with coupling to existing data sources (SCADA, weather, operational...), data cleaning and treatment.
4. `Analysis and Reporting`: predictions, trend and event detection, real-time reporting through online dashboards.

As such Zensor takes end-to-end responsibility in monitoring the health and efficiency of structures and processes, and we will develop these aspects in the following paragraphs.

$$Hardware \rightarrowtail\!\!\!\!\longrightarrow Installation \rightarrowtail\!\!\!\!\longrightarrow Management \rightarrowtail\!\!\!\!\longrightarrow Analysis$$

Figure 2.1: Building blocks for full project

Hardware and/or other Existing Data sources Deriving valuable insights from a monitoring system states from the data:

A identifying and locating the relevant data in existing databases/data warehouse;

B putting the right sensors, with appropriate settings, on the right positions and measurement conditions and, afterwards, reading them out in the optimal way;

Option A, B or both are available: it depends on whether enough data is available from the start. All this is clearly defined in each asset-specific package.

Installation and Commissioning Initially links to the existing data sources are established and data gets ingested. Where required a set of acquisition units and sensors is installed on the machine or structure. After a final verification on the spot (SAT) the monitoring system is launched: the assets enter the Internet Of Things.

Data Management Data is continuously streaming in from individual setups as well as historian sources. Structuring, verifying and cleaning the data sets is an essential prerequisite to allow for a profound analysis afterwards: on your way to an automated, continuous and smart follow-up.

Analysis and Clear Reporting Advanced insights are unlocked using algorithms based on physics as well as big data approaches, as seen in Chapter 1.1 Clear dashboards, warnings and periodic reports inform the owner or line manager about the present state and upcoming issues. Surprises are avoided, standstills reduced.

Flexibility

Now that we are clear about the possible components of the *Service as a Service* (SaaS) [20], we can distinguish two possible different types of operational flow, as illustrated in Figures 2.1 and 2.2.

$$Datasources \rightarrowtail\!\!\!\!\longrightarrow Commissioning \rightarrowtail\!\!\!\!\longrightarrow Management \rightarrowtail\!\!\!\!\longrightarrow Analysis$$

Figure 2.2: Building blocks for light project

The reasons for having two paths with separate initial phases are several. One important factor is the current global semiconductor and chip shortage [21], which makes it difficult to procure *Industrial Personal Computers* (IPCs) and sensors. On the other hand the light package (Figure 2.2) could represent a first step, to try the service with a limited effort. Having said that, it has lower costs and timelines compared to the full package, but it complicates multi-source data management and the analysis that can be performed will be of lower granularity. Table 2.2 provides a further comparison.

Full	**Light**
The engineering team has to design and devise which sensors to use, where to place them and why.	Some sensors/devices are already in place and can be integrated into the service
Hardware initial cost is high, and it requires weeks or months before it is collected and delivered to the Zennor offices.	Possibility to manage client's independently metric collected data, with an additional effort in the third step
Installation phase requires several steps, including testing hardware and connection both at the factory (FAT) and on-site (SAT)	Necessary and important information, such as network protocol and connection information, to set up links to existing data sources are collected.

Table 2.2: 1° `Hardware` vs `Datasources` and 2° `Installation` vs `Commissioning` Phases differences

That said, to give you some sense of what the service might do, here is a non-exhaustive list of the main advanced monitoring features and metrics available:

- `Availability`: have a continuous idea of availability, automatically as the platform combines different input streams and contextual information.
- `Performance`: based on the data collected and machine-learning based

methods for determining the operational condition the performance is calculated.

- `Warnings`: whenever values start to deviate, or data streams stop, warnings are sent. This avoids "black holes" in the insights of the production line or assets.

- `Quality`: coupling to existing databases or using human input fields the product quality is linked to operational process parameters.

- `Factory information systems`: such systems are crucial for obtaining operational excellence. When well managed they maximize efficiency and effectiveness. Automated data collection and advanced analysis makes this possible.

Maintenance metrics

A typical analysis involves performing a variety of calculations to obtain useful metrics, from static to maintenance; but what are the maintenance metrics anyway? There are two categories of maintenance KPIs which include the leading and lagging indicators. The leading indicators signal future events and the lagging indicators follow the past events. We will give space to the latter category. For completeness here is how Infrastructure Analytics platform could help you, asset owner, dealing with $MTTF$, $MTBF$, OEE.

- ⋆ $MTTF$: The *Mean Time Till Failure* is tracked continuously, for each asset covered the overall "disturbance free" operation is displayed.

- ⋆ $MTBF$: As events and operating conditions are automatically detected the *Mean Time Between Failures* is determined continuously, giving a good insight on where optimization is possible.

- ⋆ OEE: Using all parameters cited above the *Overall Equipment Effectiveness* is determined, a major parameter for optimizing asset management strategies and future investments, with a huge cost savings potential.

2.3 Workflow

As we will see better in Chapter 4 dedicated to practical use cases, where we will give more space to some technical details, my internship, for reasons of limited time and domain knowledge, has focused more on the last two phases, out of the four, of Zensor's product discussed previously in Section2.2. Now, in this last section, I would like to present some peculiar elements and techniques that Zensor employs for the 3rd and 4th phases: data management and analysis.

Architecture

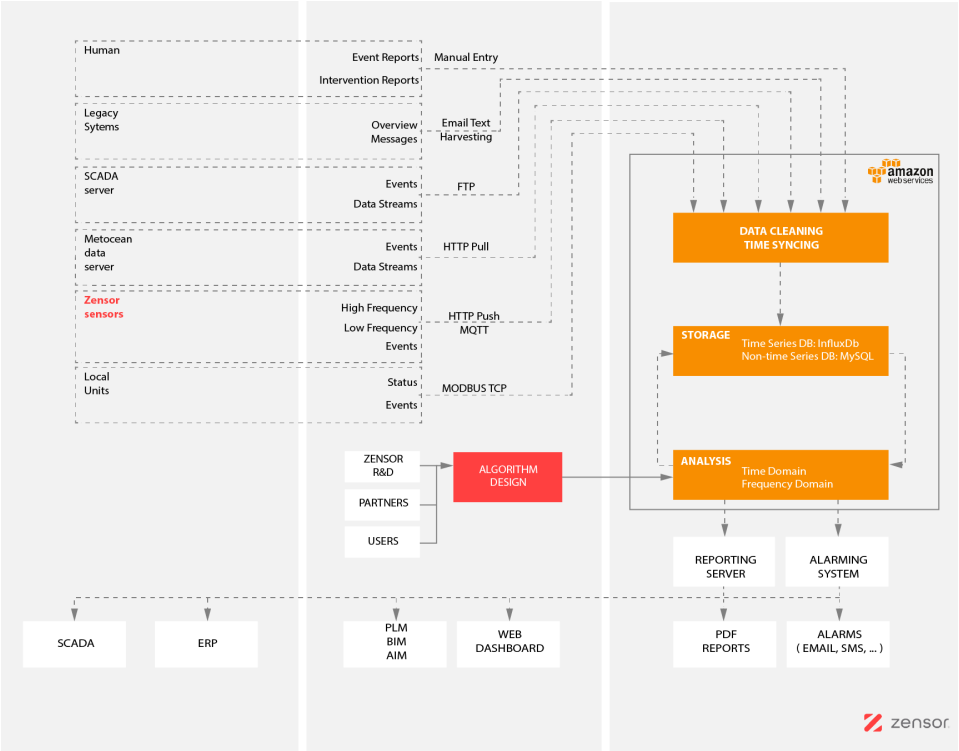


Figure 2.3: Zensor's system architecture for data analytics

Making the assumption that the first two steps have gone well, we make a simplified scenario: the sensors of project *22001_XXX* have been properly installed and wired to the IPCs, programmed by the engineering team to collect data and aggregate it into files. We could say that the "pump" from the data lake[1] is up and running, but now the "pipelines" need to be built. As interesting a discussion as this is, this is likely to get off-topic. To stay at a shallow level let's take a look at Figure 2.3; it shows us that:

- The possible data sources, including our sensor data, are various, not to mention the formats or what protocols are used for managing transfer.
- All "roads" lead to *Amazon Web Services* (AWS), ensuring that each client will then have its own separate and independent cloud EC2[2] virtual machine, with its own domain. Once data reach this destination a sequence of operation may happen:

[1]Type of data repository that stores large and varied sets of raw data in its native format: https://www.redhat.com/en/topics/data-storage/what-is-a-data-lake

[2]Amazon Elastic Compute Cloud allows users to rent virtual computers on which to run their own computer applications.

1. raw data gets cleaned, time synced and ingested in *Simple Storage Service* (S3)[3] as backup system;
2. most of the continuous analysis, also designed and implemented by Zensor R&D, is performed;
3. multiple services are running, like database and web dashboard instances, see InfluxDB 3.2 and Grafana 3.3 respectively.

- From there some advanced feature are deployed: for instance an alarming system, with SMS and email notification, or PDF reporting tool for "freezing" in time some of the most interesting dashboard panel with custom-made extra insights.

Zensor Library

Zensor library is an in-house developed Python package, containing functionality which is generic and useful to a number of projects. At present, the library contains the following main modules: clients, alerts and time.

Clients Influx client: easy interface to InfluxDB, built with the aid of Pandas 3.1, for read/write operations from/to the database and for database management operations, like showing list of measurements or dropping a measurement. Same goes from MySQL/Postgres client, that, at the moment of writing, has less functionality.

Alerts Three submodules are available:

- Heartbeat: send a heartbeat (a lightweight daemon to periodically check the status of your services) from a server via the OpsGenie platform indicating that server and scripts docker are functioning well and are available.
- SMS: send an alert directly to list of phone numbers, with the downside that long messages will be cut off at 160 characters, but phone numbers aren't terribly format sensitive and BE prefix +32 is automatically added, if no country code is specified.
- Email: This will send an E-mail through the AWS Simple E-mail service where messages sent from EC2 instances are free up to 62,000 per month.

Time This includes two functions, designed to ease writing scripts that generate reproducible time ranges, ensuring that analysis is always aligned on the correct time window:

[3]AWS's Object-Storage service: it is not a file system, instead, is a 'Key-Value' store.

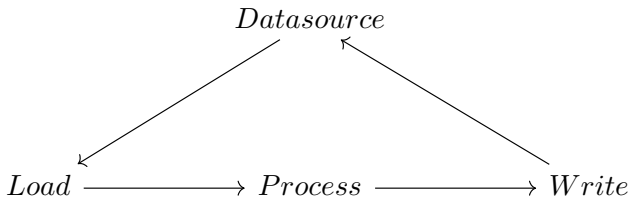


Figure 2.4: Analysis script structure

1. first one will generate evenly spaced timestamps in a given range;
2. the second one, instead generated pairs of times, which are separated by the argument (time) `interval`, the "amplitude" of the window.

The goal of a good script should be that this happens reliably regardless of **when** exactly the script is run. In this prospective, *generate_timewindow* can be useful e.g. to generate a window for the current year/month/week to date.

Script Structure

Most scripts that run on the Zensor platform have a very common structure 2.4, so for a given time window, they:

1. Load batch of data (either raw or from InfluxDB).
2. Process it in some (clever!) way, e.g. computing a derived metric.
3. Write the results out to InfluxDB, to be shown in a dashboard.

What time window they operate on will depend on what the task is, but also on whether the script is being invoked automatically by cron, a job scheduler on Unix-like OS, or manually. If a script is being invoked *manually*, this is usually to run it over historical data (e.g. rerunning a script for the month of February 2022). We typically call this operation **backfilling**. Typically, if the script is running in *cron*, it's loading "recent" data (e.g. from the past hour or past day) ending at the time the script started. Scripts on the Zensor platform need to support running in both modes, so there are a few guidelines to keep in mind when writing code.

Chapter 3

Tools for data analysis

The idea of this chapter is to show the main tools, part of Zensor's software stack, used during my work experience, in particular for performing daily data analysis task, as described in Section 2.3. This chapter is divided into three parts, each focusing on a specific pieces of software. Each developer tool covers one or more phases of the data analysis process, as explained in subsection 1.1.1:

1. Data processing & cleaning: Pandas (with Python)
2. Data storage & warehouse: InfluxDB
3. Data visualization & exploration: Grafana

These three softwares are interesting both taken individually, as we will see shortly, and together, as we will discuss in more detail in the next chapter 4, with some practical examples.

3.1 Pandas

Pandas is a data manipulation and analysis software library created for the Python programming language. It provides data structures and functions for manipulating numerical tables and time series. It is free software distributed under the BSD three-clause licence [22]. The name derives from the words "panel data", which is an econometrics term for data sets that comprise observations for the same individuals over several time periods and, at the same time, is a parody of the term "Python data analysis" [23].

3.1.1 Series and DataFrame

Pandas is primarily used to analyse data. It supports data import from a variety of file formats, including *Comma-separated values* (CSV), JSON,

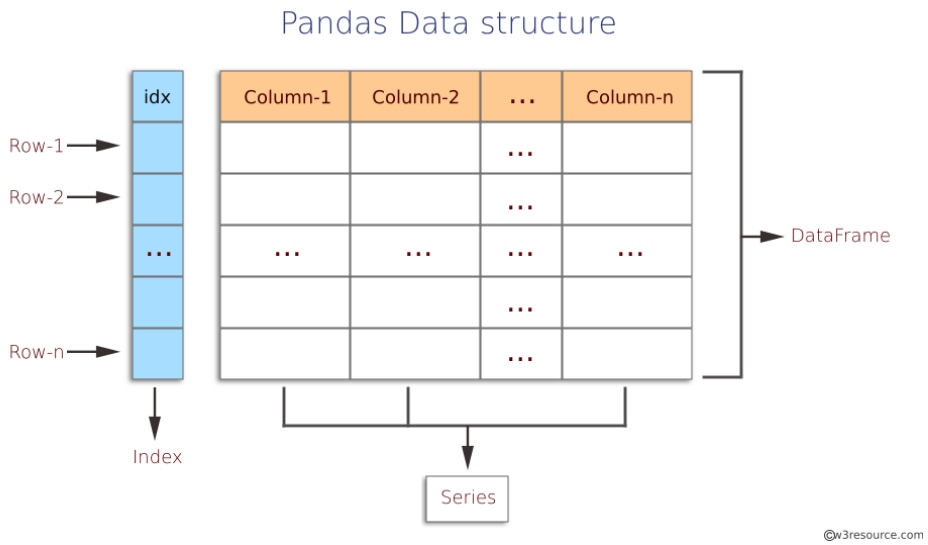


Figure 3.1: DataFrame: Pandas' core data structure

SQL database tables or queries, and Microsoft Excel spreadsheets [24]. Furthermore, Pandas supports a variety of data manipulation operations such as merging, reshaping, and selecting, as well as data cleaning and handling. To accomplish these goals, Pandas defines and makes use of two important software *Classes*, Series and DataFrame, which we will now briefly introduce.

Series Is a one-dimensional labelled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the *index*.

DataFrame Is a 2-dimensional labelled data structure with columns of potentially different types; in some ways it is similar to a spreadsheet or an SQL table (see Figure 3.1) in which each individual column is a *Series*. It is, generally, the most commonly used pandas object, since it accepts different kinds of input, which makes it very flexible [25]. Along with the data, user can optionally pass index (row labels) and columns (column labels) arguments. By doing so, the user guarantees the index and/or columns of the resulting DataFrame. If axis labels are not passed, they will be constructed from the input data based on common sense rules. And this is just one way of building a DataFrame, an appetizer of the flexibility of this library.

3.1.2 Core Features

The goal of this subsection is to give an outline of how many possible use-cases are covered by this library, and, at the same time, to explore a

couple of them that proved to be crucial during my internship experience. We are now first illustrate the concept of *grouping*.

Groupby The name **GroupBy** should be quite familiar to those who have used a SQL-based tool or worked with relation database. This "engine" allows split-apply-combine operations on heterogeneous data sets. By "group by" we are referring to a process involving one or more of the following steps:

1. Splitting the data into groups based on some criteria.
2. Applying a function to each group independently.
3. Combining the results into a data structure

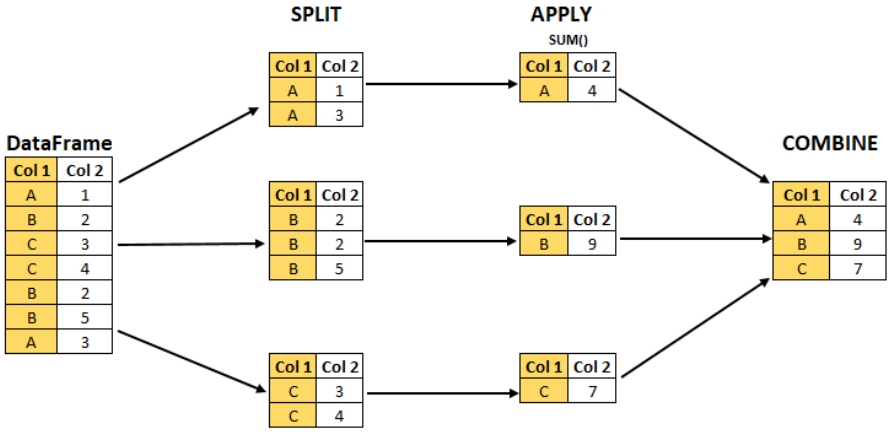


Figure 3.2: Shows the Split-Apply-Combine using an aggregation function (Source: [26])

Among these three steps, the *split* is the most straightforward. In fact, in many situations, we would like to split the data set into groups and do something with those groups [25]. In the *"apply and combine"* step, we might wish to do one of the following:

- Aggregation: compute a summary statistic (or statistics) for each group, some examples:
  - Compute group sums or means.
  - Compute group sizes / counts.
- Transformation: perform some group-specific computations and return a like-indexed object, for instance:
  - Standardize data (zscore) within a group.

  - Filling NAs (value that are not valid) within groups with a value derived from each group.

- Filtration: discard some groups, according to a group-wise computation that evaluates True or False, like:
  - Discard data that belongs to groups with only a few members.
  - Filter out data based on the group sum or mean.

- Some combination of the above: **GroupBy** will examine the results of the *apply* step and try to return a sensibly *combined* result if it doesn't fit into either of the above two categories.

Since the set of object instance methods on pandas data structures are generally rich and expressive, we often simply want to invoke, say, a DataFrame function on each group. With this engine you can try multiple different approaches, testing what suits more your needs, even though often it is hard to define these three separate steps for badly shaped data.

Time series resampling Pandas contains extensive capabilities and features for working with time series data for all domains. Using the **NumPy** datetimes dtypes, it has consolidated numerous features from other Python libraries (like **scikits.timeseries**) as well as created a tremendous amount of new functionality for manipulating time series data. As an example, Pandas supports:

- Parsing time series information from various sources and formats
- Manipulating and converting date times with timezone information
- Moving window statistics and linear regressions

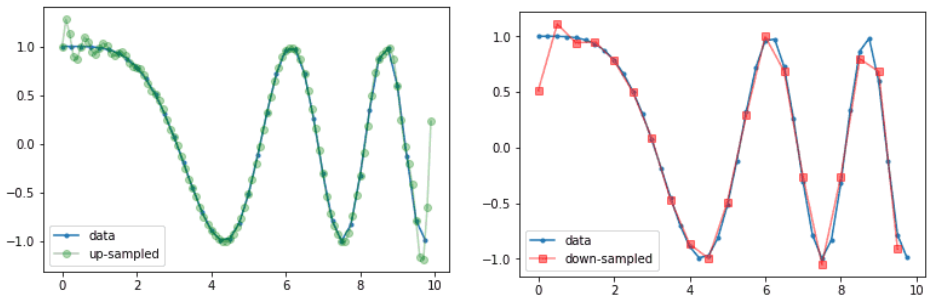


Figure 3.3: Examples of signal waveform data processed by resample

Let us now focus on one key aspect of handling/manipulating sensor data, crucial in our context, the necessity of resampling. Fortunately, Pandas has,

once again, a simple-to-use, powerful, and efficient functionality for performing resampling operations during frequency conversion (e.g., converting secondly data into 5-minutely data). This is also extremely common in, but not limited to, financial applications [24].

To keep things simple, we could say that resample is a time-based **Groupby** followed by a reduction method on each of its groups; as a positive side, this method can be used directly from *DataFrameGroupBy* objects that we discussed in paragraph **GroupBy** previously discussed. The resample function is very flexible and allows you to specify many parameters to control the frequency conversion and resampling operation, both up sampling and down sampling.

Others functionality Furthermore, other time series features are available, and not only that notably:

- Date range generation and frequency conversions
- Data alignment, shifting and lagging
- Integrated missing data handling
- Data set reshaping, pivoting, merging and combining
- Label-based slicing, sophisticated indexing, and big data set sub-setting
- Insertion and deletion of columns in a data structure.

Conclusion Pandas provides a solid foundation upon which a very powerful data analysis ecosystem can be established, especially since the library is performance-optimized, with important code paths implemented in C and Cython[1].

3.2 InfluxDB

Time Series In mathematics, a time series is a series of data points which are indexed (or listed or graphed) in time order (see 3.4). More generally, a time series is a sequence taken at evenly spaced intervals over a period of time. As a result, it's a sequence of discrete-time data. Ocean tidal heights, sunspot counts, and the Dow Jones Industrial Average's daily closing value are all examples of time series.

[1]Optimistic static Python compiler capable of using highly optimized C libraries https://cython.org/

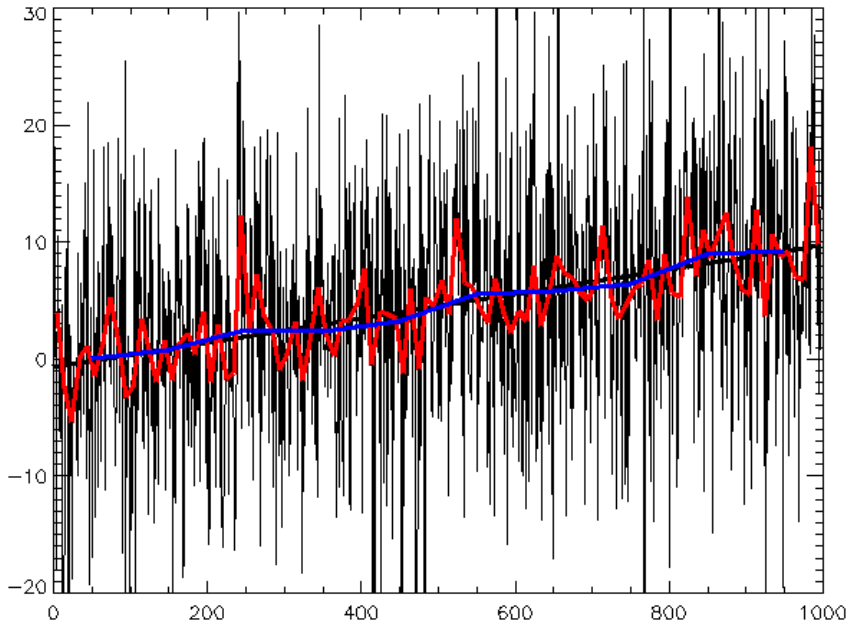


Figure 3.4: Random data plus trend, with best-fit line and different smoothing applied. (source: Wikimedia Commons [27])

3.2.1 TSDB: time series database

It follows that a time series database TSDB is a software system that is designed to store and serve this peculiar type of data, time series, using time(s) and value(s) pairs(s). Timescale, popular TSDB, CEO *Ajay Kulkarni* [28] put it:

> Time-series datasets track changes to the overall system as INSERTs, not UPDATEs. This practice of recording each and every change to the system as a new, different row is what makes time-series data so powerful. It allows us to measure change: analyse how something changed in the past, monitor how something is changing in the present, predict how it may change in the future.
>
> So here's how I like to define time-series data: data that collectively represents how a system/process/behaviour changes over time.

Although it is possible to store time-series data in many diverse database types, the design of these systems with **time** as a **key** index is distinctly different from relational databases, which reduces discrete relationships through referential models. In many cases, the repositories of time-series data will utilize compression algorithms to manage the data efficiently [29]. Furthermore, time series databases can also be configured to regularly delete old data,

unlike traditional databases which are designed to store data indefinitely.

3.2.2 Influx solution

InfluxDB is an open-source time series database TSDB created by the InfluxData organization [30]. It is written in the Go programming language and is used for time series data storage and retrieval in various sectors such as operations monitoring, application metrics, Internet Of Things, and, especially important in our context, sensor data and real-time analytics. It also supports the processing of data from Graphite, a data logging and graphing tool for time series data [31].

Core Features InfluxDB has no external dependencies and provides a SQL-like vocabulary with built-in time-centric functions for querying a data structure made up of measurements, series, and points, which listens on port 8086 [32].

Time	location	scientist	butterflies	honeybees
2015-08-18T00:00:00Z	1	langstroth	12	23
2015-08-18T00:00:00Z	1	perpetua	1	30
2015-08-18T00:06:00Z	1	langstroth	11	28
2015-08-18T05:54:00Z	2	langstroth	2	11
2015-08-18T06:00:00Z	2	langstroth	1	10
2015-08-18T06:06:00Z	2	perpetua	8	23
2015-08-18T06:12:00Z	2	perpetua	7	22

Table 3.1: Sample time series dataset: number of butterflies and honeybees counted by two scientists

Each point is made up of a fieldset and a timestamp, which are key-value pairs. These pairs form a series when they are grouped together by a set of key-value pairs known as a tagset. Finally, a measurement is created by grouping series together using a string identification. 64-bit integers, 64-bit floating points, strings, and booleans are among the possible values as shown in Table 3.1. The time and tag set are used to sort the points. As a side note it is important to know that data is downsampled and removed according to retention policies, which are set by measurement and that *Continuous Queries* are executed on a regular basis and the results are stored in a goal measurement.

Design Tradesoff InfluxDB is a time series database and, by this aspect, has several limitations; as a matter of fact, being optimized for this use case involves a number of trade-offs, primarily to increase performance at the cost

of functionality [32]. Below, it is reported a list of three design ideas that lead to compromises that I personally encountered during my experience:

1. Deletes are a rare occurrence. When they do occur, it is almost always against large ranges of old data that are cold for writes.

2. Updates to existing data are a rare occurrence, and contentious updates never happen. Time series data is predominantly new data that is never updated.

3. Many time series are ephemeral. There are often time series that appear only for a few hours and then go away, e.g., a new host that gets started and reports for a while and then gets shut down.

Pros	Cons
Restricting access to deletes and updates allows for increased query and write performance	Delete and Update functionality is significantly restricted, since InfluxDB is not CRUD
InfluxDB is good at managing discontinuous data	Schema-less design means that some database functions are not supported e.g. there are no cross table joins

Table 3.2: Pros and cons of InfluxDB

Conclusion It is therefore not surprising to conclude that, when compared to a general purpose relational database like SQL Server, InfluxDB, using default single node configuration, outperformed both write speed, disk storage usage (by a factor of 27x) and query execution time, where InfluxDB is up to 20x faster with an average of 8x faster [33]. It can be seen that InfluxDB, tailored-made for Time Series data, is released after other competitive technologies (like Graphite, TimescaleDB, Prometeus) and yet still 1st among the popularity tier list[2]. The InfluxQL, SQL-subset query language, helps make it easier to use and adapt for people who are used to working with relational databases such as MySQL, even though influx implementation is a smaller subgroup, not fully supporting *Create Read Update Delete* (CRUD) operations. For more complex queries is almost mandatory using Flux, a functional-like query language developed in-house by Influx [32].

[2] https://db-engines.com/en/ranking_trend/time+series+dbms

3.3 Grafana

Grafana is a web-based analytics and interactive visualization application that runs on a variety of platforms. When connected to supported data sources, it produces web-based charts, graphs, and alerts. Grafana Enterprise, a paid version with more features, is available as a self-hosted installation or as a Grafana Labs cloud service account [34]. Grafana is split into two parts: a front end and a back end, both of which are built in TypeScript and Go. Because the project is open source and has a large existing community, it can be expanded through a plug-in system, adding specific customizations to the existing platform [35]. Using interactive query builders, end users can develop complicated monitoring dashboards.

But what is a dashboard anyway?

3.3.1 Dashboard: what is it?

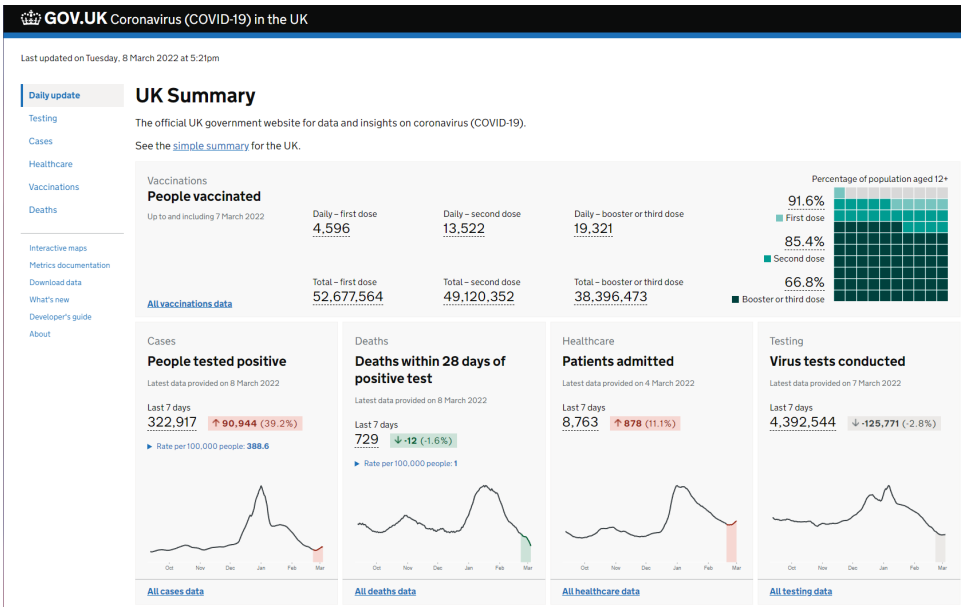


Figure 3.5: Official data and insights on coronavirus (Source: gov.uk)

A dashboard, in business, is a type of graphical user interface that often enables quick access to key performance indicators (KPIs) related to a certain goal or business activity. In our context, "dashboard" refers to a "progress report" or "report" and, as a type of data visualization, it is mostly accessible by a web browser and is usually linked to regularly updating data sources. The term dashboard is derived from the automotive dashboard, where drivers may monitor the primary functions at a glance using the instrument panel.

The success of dashboard projects depends on the relevancy/importance

of information provided within the dashboard. This includes the metrics chosen to monitor and the timeliness of the data forming those metrics; data must be up-to-date and accurate. Well known dashboards include Google Analytics dashboards, used on 55% of all websites [36], which show user activity on a website, or the UK government, and similar for each country, coronavirus tracker, for the COVID-19 pandemic [37].

An interesting project is the GLAM Wiki dashboard, from Israel [38]. Its purpose is to assist GLAM institutions (galleries, libraries, archives, and museums) in tracking the use of their free-content files that they have submitted to Wikimedia projects. Based on multiple specified indices and several time frames, the dashboard visualizes statistical data that shows the extent of exposure and usage of these public-domain assets. The collecting data, which is presented in a variety of diagrams and graphs, allows the institutions to obtain insights, discover patterns and preferences, and understand the overall impact of these free materials on the global audience of Wikimedia-project users.

3.3.2 Key strengths

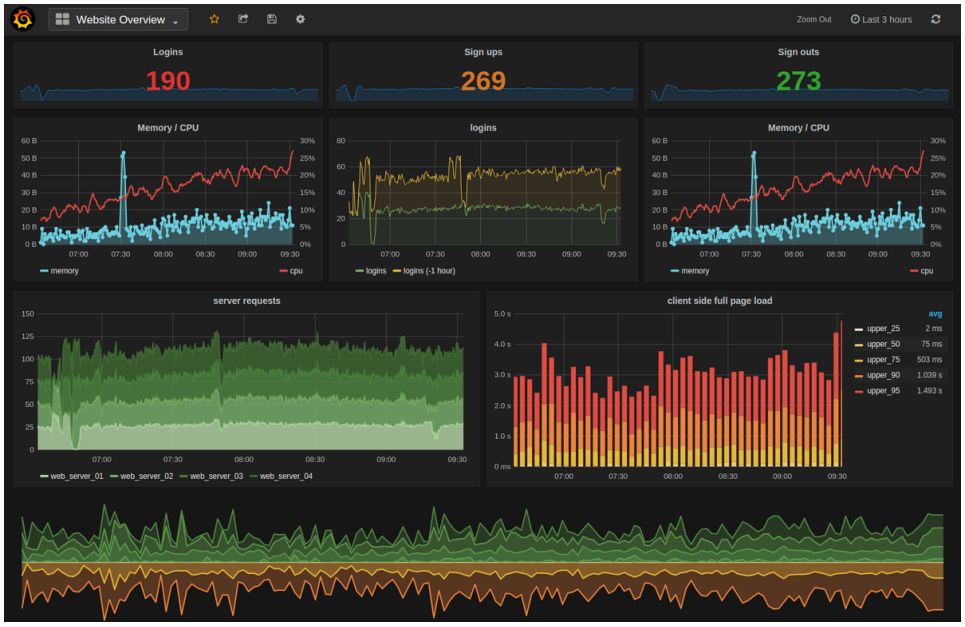


Figure 3.6: Grafana Graph Visualization (Source: Flickr [39])

Now that we are familiar with the "dashboard" concept, we can highlight why we should use Grafana solution 3.6; here are some of the key features [40]:

- **Visualize** fast and flexible visualization with a variety of options allows data to be displayed the way the user wants it;

- **Dynamic Dashboard** dynamic and reusable dashboards may be created using template variables.

- **Explore Metric** ad-hoc queries and dynamic drill-down permits for data discovery. View splits and side-by-side comparisons of different time ranges and data sources is easily achievable;

- **Explore Logs** fast switch from metrics to logs preserving label filters. Furthermore, searching the logs is rather quick and can be performed on live streams;

- **Alerting** most of the vital/operational metrics may be alerted visually and different types of notifications (SMS, mail, Slack) may be despatched with the aid of Grafana;

- **Mixed Data Source** the same chart can have different data source: these can be selected based on queries, with built-in support for most of the prominent data sources available in the market as well as custom ones;

- **Annotations** graphs may have events that can be annotated, two solutions are possible: use native annotation store, with the ability to add annotation events directly from the graph panel or via the HTTP API, or querying other data sources. Event metadata and tags can be seen when hovering over events.

Time ranges Grafana provides several ways to manage the time ranges of the data being visualized, for dashboard, panels and also for alerting, with the support for following time units: s (seconds), m (minutes), h (hours), d (days), w (weeks), M (months), Q (quarters) and y (years).

Example relative range	From:	To:
Last 5 minutes	$now - 5m$	now
The day so far	now/d	now
This week	now/w	now/w
This week so far	now/w	now
This month	now/M	now/M
This month so far	now/M	now
Previous Month	$now - 1M/M$	$now - 1M/M$
This year so far	now/Y	now
This Year	now/Y	now/Y

Table 3.3: Time units and relative ranges example

The minus (−) operator allows you to step back in time, relative to *now* and if the user wish to display the full period of the unit (day, week, month, etc...), append /<time unit> to the end. It is also possible to view fiscal periods, using fQ (fiscal quarter) and fy (fiscal year) time units. The plus (+) operator, instead, allows you to step forward in time relative to *now*. For example one might use this feature to look at predicted data in the future. Here [3.3] you can see some relative ranges samples.

The minimum dashboard refresh interval is, per default 5 seconds; this can be increased or decreased keeping in mind that it will restrict users to set the refresh interval of a dashboard lower than given interval and anything below the default value could logically impact performance and responsiveness.

Loading speed Loading speed of a Grafana dashboard depends on 5 major things:

1. pre-selected and saved time window: the larger the queried time period, the longer it takes to open and display the contents;
2. data frequency in the panels: in case of the very high frequency, non aggregated data, even if selected time period is minutes, it will take time to load;
3. the number of panels with the data inside;
4. your database structure;
5. whether calculations have to happen inside the panel before the data is displayed.

Conclusion Grafana is the right choice when visualizing infrastructure, applications, network devices, sensors, and more. This is a great 24/7 monitoring solution for NOC and DevOps teams. It can also help to manage all data from other application monitoring tools like AppDynamics, New Relic, Splunk, Dynatrace and all-in-one web interface for data viewing, alerting, and reporting. A further comparison with other data visualization tools, such as Power BI and Tableau, might make interesting reading.

Chapter 4

Use cases of interest

1.DG: Intro This chapter is about ...

4.1 Analyse blade grinder vibration

In this section, we discuss how to improve an existing industrial machine, needed for blade production. The three years old, installation has a high number of standstills (unplanned stop) with a strong impact on sales. Furthermore, the excessive amount of vibrations on the machine has negative effect on quality of the cut, invalidating quality test results.

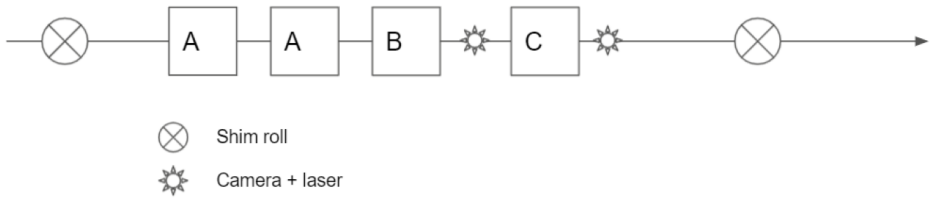
We are also going to illustrate some difficulties encountered in vibration analysis and correlated problems, where the full understanding of problem's context demonstrated to be crucial. Despite the lack of precise operational data, counter-intuitive results, the initial report was successful and helped to strengthen the methodology for further analysis.

4.1.1 Initial Hypothesis

Context Introduction The subject of this monitoring operation is a blade-cutting machine line. As shown in Figure 4.1 there are four different stations (a). Each of them has one or two motors type A, B, C (b). Each station has grinding stones to sharpen a blade, which is one long strip of steel passing through all the machines. To perform their grinding task, the stones turn around their axe sharpening the steel blade, each machine in its own way.

Client Stumabo International, a manufacturer of precision blades for the food processing industry, produces several million blades per year, and it is known in the industry for their progressive industrial potato cutting solutions, and innovative shapes produced through hydro-cutting systems. Stumabo also uses their knowledge to integrate the best blade in the FAM industrial

(a) Line overview: side view



(b) Line motor schematics

Figure 4.1: Stumabo's blade production line covered by this project

mechanical cutters [41].

4.1.2 Goal(s), purpose & critical factors

The project started in conjunction with the beginning of the internship (Oct-Nov 2021), and I had the pleasure of contributing in its early stages. The main objective was to produce an *ad-hoc*[1] analysis report, that could answer specific business questions. Unfortunately my practice ended before the second, more substantial monitoring phase of the project began, scheduled to be April 2022. Let's see what the goals are in the long and short term and then exploring the latter.

Long term: project lifecycle

- ◎ Increase production quality through the use of continuous data analysis.
- ◎ Avoid unplanned standstill (downtime) by preventing critical component failures.
- ◎ Extends the life of machines and installation through Predictive maintenance and Condition monitoring (see Section 1.2)

Short term: *ad-hoc* campaign

[1] It is usually produced only once, is more visual than a static report and, once completed, is shared with a smaller audience.

- Is it possible to identify a strong impact of the turning of the grindstones? Perhaps a possible imbalance?
- As a general insight, do we have indication that something is causing strong vibration?
- Lastly, as preliminary path for the second stage, the *Continuos monitoring* (CtM), which key elements we should focus on? **2.EB: questo punto non l'ho capito, DG: meglio?**

4.1.3 Project description by phases

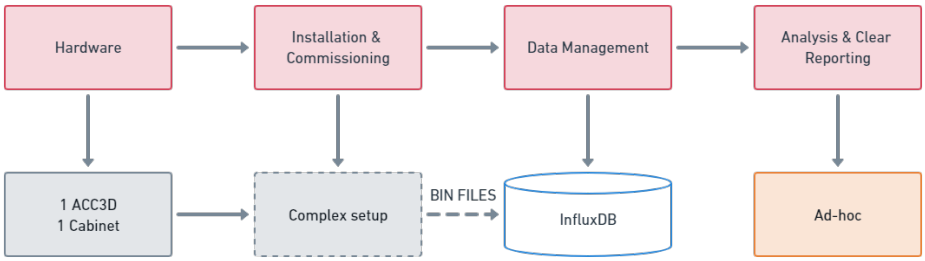
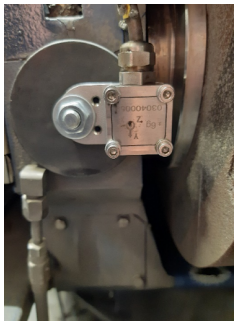


Figure 4.2: Stumabo's (full) project core stages

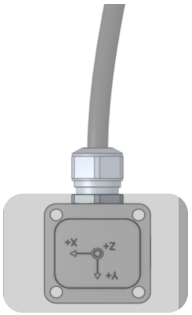
As we saw earlier in Section 2.2, we can have two types of projects, *full* and *light*. This one belongs to full or "complete" class, with some intense engineering (hardware & installation) phases. **3.Perché less complex than big project? DG: perchè è una campagna di preparazione inziale.**
4.E cosa è un "big" project? big confonde, meglio full. Magari era scrtto prima. Ma bene sia ricordarlo che aggiungere riferimento a "prima" DG: il riferimento era già presente ad inizio paragrafo, ne aggiungo un' altro?

However, since this is a preparation (*ad-hoc*) campaign, as mentioned earlier, it will have less complexity than a regular *full* project. To summarize it in one formula, given three variables ($f$ull, $l$ight, $s$tumabo) and "complexity" function $O(x)$ is true that $\forall f, l | O(l) \leq O(s) \leq O(f)$. My contribution was to help with the analysis phase, but it's important to emphasize how the first three phases went, because they will have serious implications for data development and analysis.

Hardware One 3-dimension accelerometer, called *ACC3D* in Figure 4.3c, and a "mobile cabinet" was provided to Stumabo, which they placed on each of the four machine and roughly kept track of the different positions and operations in a log file. Unfortunately, this is the only operational data available, with the following structure (see Table 4.1). As it can be seen, despite being in Flemish, it has several tags but with one big issue: start and



(a) Installation photo



(b) CAD render

(c) An industrial ACC3D

Figure 4.3: Station 1 blue: sensor setup

end time are approximate and do not reflect the data trends collected by the above-mentioned sensor.

Datum	Start tijd	Stop tijd	Actie	Opmerkingen	Opstelling
Date	*Start time*	*Stop time*	*Action*	*Notes*	*Setup*
08/10/2021	13:19	13:29	Test	Enkel ...	grote ...
...	...	...	...	...	...
30/11/2021	12:35	14:07	Prod	...	...

Table 4.1: Stumabo's log file structure

Installation We must therefore clarify, before we go any further, the industrial setup: in our aid, come the engineering schematics showed in figure 4.4; it consists in three subfigures: the first one (a) represents the same production line we saw earlier at 4.1a, just from another prospective. This view from above makes it easier for us to identify some important elements, e.g. how many stones are present for each station. It is essential to clarify a simple convention: within Stumabo the right side of the line is referred to as red, *rood* in Flemish, and, in the same way, the left side will be referred to as blue, *blauw*. Speaking about colours, we can now make a direct connection with Figure (b), which shows us the evolution of the blade during the process. First it is sharpened, from the 1st blue station, on the left side, and then it passes to the 1st red station, where it will be rolled on the right side. Same goes for 2nd and 3rd station, that have two grinders aligned.

Now that we are aware of how the industrial process works, let's move on to diagram (c). It represents in more detail how the sensor has been installed and what consequences it has with respect to the coordinate system, a somewhat complex issue that has caused several doubts and discussions

in the analysis phase. This would therefore appear to be a good point to take stock of the situation. The accelerometer was installed within the line, adjacent to the stones and close to the blade, by means of a round magnet, as shown in Figure 4.3a. Moreover, the same device was placed, at different times in different stations, mirrored on both sides, except the last one, for a total of five different locations: $S_1R\&B$, $S_2R\&B$, S_3B as shown in the bottom part of Figure 4.4a.

It follows that **two** different **reference systems** are present:

- local (x, y, z) for each sensor placement;
- global (X, Y, Z) for the entire production line.

We can see some refences of this important aspect in the mentioned figures alongside. This dual model well abstracts the reality, but does not allow a comparison between the various stations, which instead we would like to carry out to achieve the initial goals. So it will be necessary to transform everything in global coordinates, taking into account two additional factors: *sensor orientation* and *installation angle*.

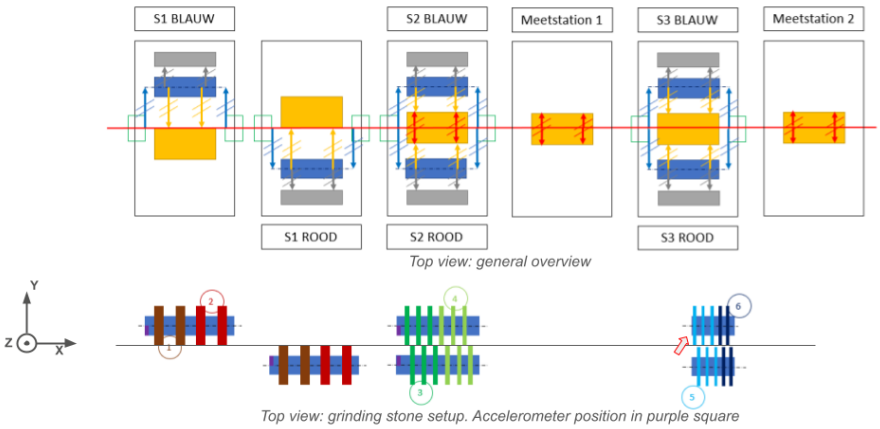
- **Sensor Orientation**: going back on Figure 4.3, we can see how the sensor has been installed horizontally, with the x-axis as parallel as possible to the blade (X) with the help of a level. **5.EB: check English: "tolerates"? DG: esiste ed è usato** This tolerating some errors due to disturbing factors such as colour, dirt and imperfections of the metal. Furthermore, the latter is installed upside down, i.e. parallel to the Z axis as shown in the Render 4.3b.
- **Installation Angle**: We had to take also into account that the machines have various angles to the ideal Z-axis, this was measured on site and varies from station to station, from 5° to 12° degrees.

We will delve into both aspects in the final stage of analysis, but now we would like to clarify following equation-mapping (4.1) from *local* to *global* coordinate. Here $\widehat{widehat}$ stands for the *rotation* operator and $\underline{underline}$ for *transposition*.

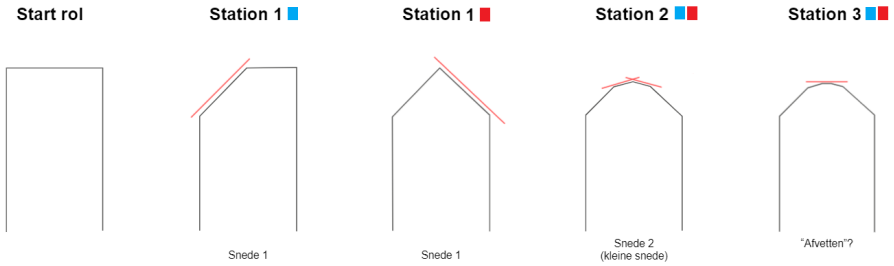
$$\begin{cases} x = \alpha & \Rightarrow \; x = \widehat{\alpha} & \Rightarrow \; X = \underline{\widehat{\alpha}} \\ y = \beta & \Rightarrow \; y = \widehat{\beta} & \Rightarrow \; Y = \underline{\widehat{\beta}} \\ z = \gamma & \Rightarrow \; z = \widehat{\gamma} & \Rightarrow \; Z = \underline{\widehat{\gamma}} \end{cases} \tag{4.1}$$

We now move on to the following phase.

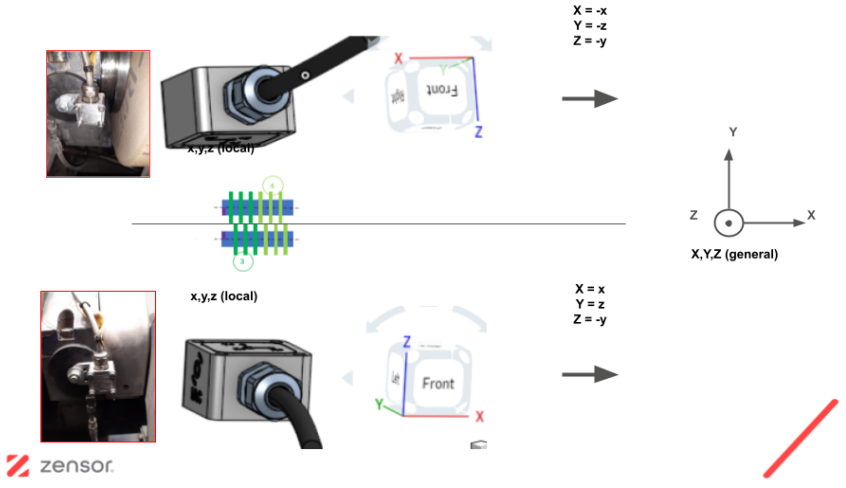
Data Management As stated previously a single sensor was placed in different locations, remaining on for the duration of the ad hoc project. We will therefore have a single data stream. The data flow is as follows: from the sensor to the cabinet to AWS and, finally, InfluxDB.



(a) Top view overview of stations plus grinding stones detail. Purple dots on the bottom details are where the $3D$ accelerometer was placed

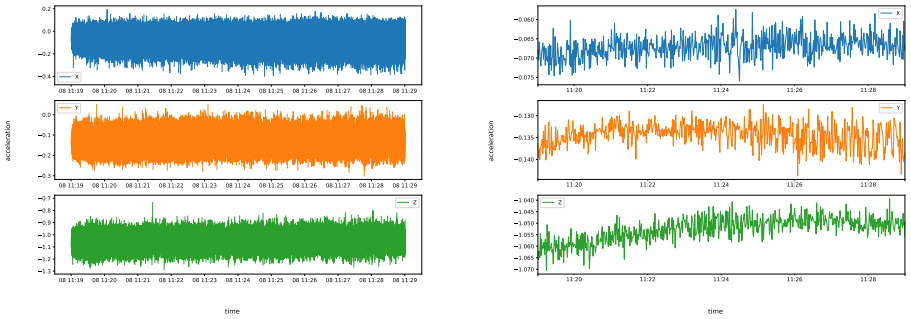


(b) Blade evolution, a sketched overview of the cut by station



(c) From local to general coordinate system, top view

Figure 4.4: Stumabo's engineering schematic, also relevant during analysis



(a) `60Hz` raw vibration (b) `1Hz` raw vibration

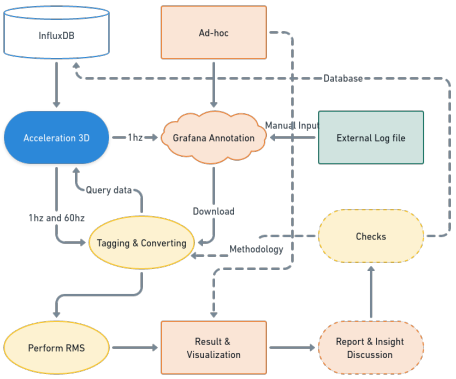
Figure 4.5: Exploratory data analysis on 1° block, station 1 blue

6.EB: mi pare che x, y, z vengano utilizati a volte minuscoli, altre maiuscoli. C'è un motivo? Spiegare motivo, oppure, se non c'è uniformare notazione
7.DG: il paragrafo precedente (installation) è dedicato principalmente a questa tematica ... As a reminder: lowercase stands for local frame of reference, while upper is for global as explained before at 4.

I would like to point out that, at cabinet level, two separate binary files are created, where x and z channels are stored together and, instead y channel is kept separated. Moreover, for each sensor channel, all vibrations are recorded at `60Hz`, or 60 points per second, for a total of 180 per second (a). As you can easily imagine when you want to visualize, e.g. doing some EDA as discussed in subsection 1.1.1, and investigate hours and hours of data these are way too many data points. This is the reason why Lambda[2], during data ingestion, aggregates the data using the mean function, and reduces the frequency to `1Hz` (b), i.e. performs a down sampling operation. It then saves both streams in the TSDB as separate measurements as shown in Figure 4.6a. It is an important element that will be useful later in analysis phase.

Analysis This is where most of my work and time went, my first task was to perform some Exploratory data analysis over the `acc_1Hz` measurement using Grafana (a). Since the cabinet was never turned off while sending data for approximately two months, except for five days, between 13/10 and 18/10, it was clear from the start that most of the signal was useless, "flat" stationary data with no movement e.g. during night hours when the servo motors were off. Another big problem was that there was no distinction whatsoever between different sensor installation positions. To solve these and other smaller issues a preliminary data labelling step was performed directly on the visualization tool, using the Excel log file as source of operational

[2]AWS Lambda is an event-driven, serverless computing platform `https://aws.amazon.com/it/lambda/`



(a) Data Management and Analysis overview

(b) Annotation process

data, with all the limitations involved. The task was performed in-place using the Annotation UI, dividing the measurement in smaller blocks of variable length as demonstrated in Figure 4.6b. While being tedious, it was strictly necessary, otherwise no meaningful analysis could have been performed.

The idea is to isolate relevant data, in other terms data that is available, and we know what it represents keeping in mind we are interested in finding sub zones where data seems to be not varying (so much): for instance we would like to exclude the acceleration and deceleration phases of servo motors. For each client log entry, we create an equivalent annotation entry, with the caveat that there could be more than one behaviour block identified per run, so we have two unique identifiers, a Block ID and a Run ID. Each block will have several variables like operation {TEST, PROD, ...}, colour {Blue and Red}, station $\{1, 2, 3\}$, and so on. These variables will be our tags columns $\{n+1, n+2, n+3, \dots k\}$ in our final result Table C. To give an approximate order of magnitude, about 110 annotations, hence blocks, were made. The next step was to download the annotations, using Grafana's REST API, and to transition to a better data structure: from intermediate JSON to structured DataFrame. Once this metadata was in place, we could proceed with the more complex section of the analytics. Once again our main concern was to study the amplitude of vibration. So for each *annotated block* we:

1. retrieved ACC data through iterative queries to InfluxDB, using start and end time;

2. manipulated it, using vector calculus, for performing rotation and translation operations;

3. returned the *Root mean square* (RMS) values for each individual axes X, Y, Z;

We are now going to take a closer look at the various operations: as far as the

first one is concerned, nothing too complicated, thanks to Zensor library we can easily retrieve our time series data in a shape of a pandas DataFrame. For the second point, here now we do not dwell too much technical details, it was agreed to combine the two operations into one (see Equation 4.1). So what we end up doing was performing dot product among vector u and matrix $R_X(\phi)$, in formula: $R_X(\phi)u = u$. With no signs assigned to the angles, that we know from engineering files, the basic translation + rotation matrix on X-axis $R_X(\phi)$ is:

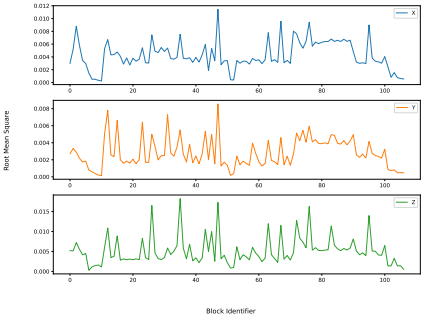
- Blue: $\begin{cases} Y &= -ysin(\phi) + zcos(\phi) \\ Z &= -ycos(\phi) - zsin(\phi) \end{cases} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & -sin(\phi) & cos(\phi) \\ 0 & -cos(\phi) & -sin(\phi) \end{bmatrix}$

- Red: $\begin{cases} Y &= ysin(\phi) - zcos(\phi) \\ Z &= -ycos(\phi) + zsin(\phi) \end{cases} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & sin(\phi) & -cos(\phi) \\ 0 & -cos(\phi) & sin(\phi) \end{bmatrix}$

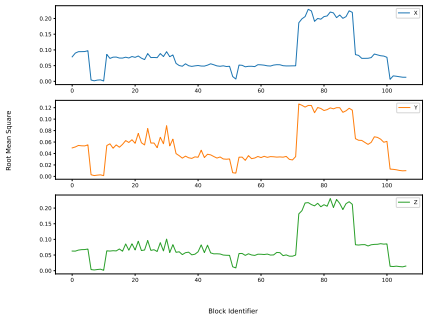
Lastly for the third step is important to point out that, for each direction, before the RMS is calculated, the average of the time-block is determined. This average will be subtracted from the array values, as to only monitor the RMS of the relative acceleration.

These three operation were combined in one python function and used alongside Pandas internals. So the methodology was firstly "called" on `1Hz` data for testing the procedure, that took approximately 15 minutes of CPU time, and then on the higher frequency data, `60Hz` where instead took more than 4 hours. At Figure 4.7 we observe three plots. For all of them we have common elements. The x-axis label is our block identifier (Block ID), while on the y-axis represent the overall RMS value of the block itself. It is quite evident how the two graphs represent similar signals but of different order of magnitude. In case a we have a $[min, max]$ range of $[0, 20] * 10^{-4}$ while in b is $[0, 25] * 10^{-2}$ which makes direct comparison difficult, see Plot c. In particular, the `60Hz` trend seems to be more constant and gradual, while `1Hz` is characterized by strong peaks. It is clear that station 2 cause most of the vibration, as we would expect, by his design with two engines closer to each other.

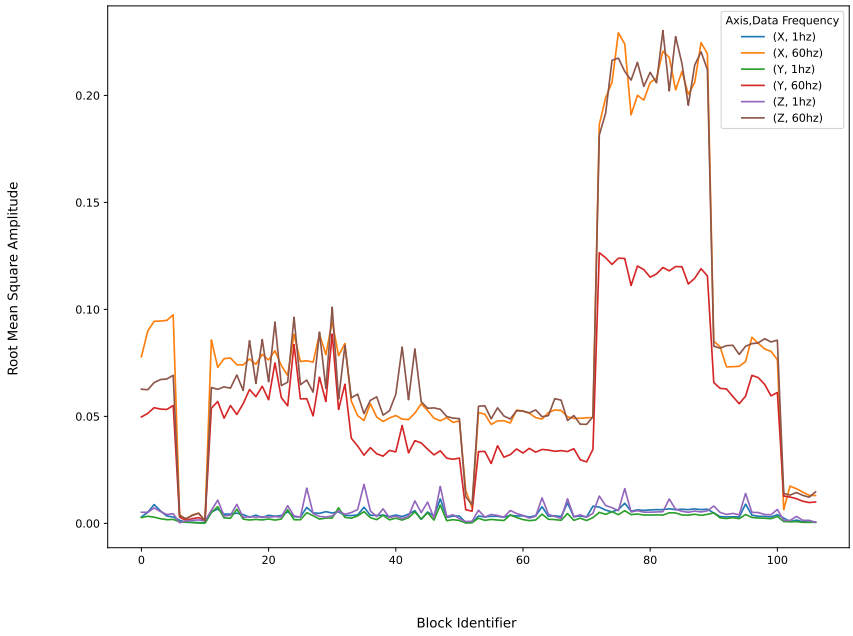
After all of these operations, we combined `1Hz` numerical values to the previously prepared metadata to achieve an interesting intermediate result that has approximately this aspect. Run ID and Block ID were removed here for relevance purposes (see Table 4.2). These results were then discussed and checked with a more experienced colleague, who had more domain knowledge. He also continued with the analysis, comparing, for example, different stations for the same type of operation. I decided not to discuss his insights, here in this report, as they are not the result of my work but a more direct consequence.



(a) `1Hz` RMS values



(b) `60Hz` RMS values



(c) `1Hz` and `60Hz` RMS values

Figure 4.7: RMS amplitude comparison between low and high frequency data

Start	einde	actie	kant	stat	draaien	X	Y	Z
13:19:01	13:29:01	TEST	B	1	1	131	176	592
13:29:01	13:39:00	TEST	B	1	1,2	160	143	461
13:38:58	13:49:01	TEST	B	1	1,2,3B	132	166	356
13:48:59	13:59:00	TEST	B	1	1,2,3R	113	149	244
13:59:00	14:08:59	TEST	B	1	1,2,3B,3R	145	143	217
14:09:00	14:20:00	TEST	B	1	1,2,3B,3R,4	176	153	294
...								
10:04:00	12:35:00	DASTE	B	3	1,2,3B,3R,4	630	542	1489

Table 4.2: intermediate result: numerical RMS `1Hz` values are given using engineering notation: 10^{-6}

4.1.4 Conclusion

Stumabo ad-hoc analysis showed, overall, insightful findings. For instance station 1 blue has higher vibration than it should, and it will be further investigated with spectral analysis for better identify the root cause. There was a less intuitive result though: vibration amplitudes (RMS) in X direction, along the blade going through the grinding stone stations, seemed more prominent than in Y direction, also horizontal but perpendicular to the blade direction. Before digging deeper into this, we wanted to check on our side if these axes can't have been switched somewhere.

The machines (motor, stones ...) and stones turning both causes vibrations. As discussed previously, for keeping temperatures low, cooling fluid is sometimes applied to the stones, where it is absorbed. When drying, we noticed higher vibrations as the liquid isn't all over the stone any more, but going to the outside gradually. Anyway, the turning would intuitively cause more vibrations perpendicular to their rotating axe (Y) and (Z), not along their axe (X), but in the analysis we saw relatively higher vibrations along the blade axe (X) than perpendicular to it (Y), even though (Z), vertically, is still highest. After successfully double-checking back-end, Lambda configuration and the *Programmable Logic Controller* (PLC) programming, we couldn't confirm 100% the cabling connection, but with a small margin of error, we can confirm that, indeed, X and Y are **not** switched. Further analysis will eventually debunk that.

4.2 Monitor electricity consumption

Meaningful introduction: such as the data collection and the target user of the dashboards or project age etc.

4.2.1 Initial Hypothesis

Client *Green Energy Park* (GEP) is a joint project of the *Vrije Universiteit Brussel* (VUB) (Free University of Brussels), dutch and English-speaking research university, and the *Universitair Ziekenhuis Brussel* (UZB) (University Hospital of Brussels), which purpose is to develop and operate a research campus in the Research park of Zellik with a focus on the following three research domains:

- Energy and mobility transition.
- Hospital of the future, part of *Brussels Health Campus* (BHC).
- Smart regions.

With this research campus, Green Energy Park aims to bridge the gap between research, innovation, realization and exploitation, by acting as a large-scale living lab, expertise and training centre [42].

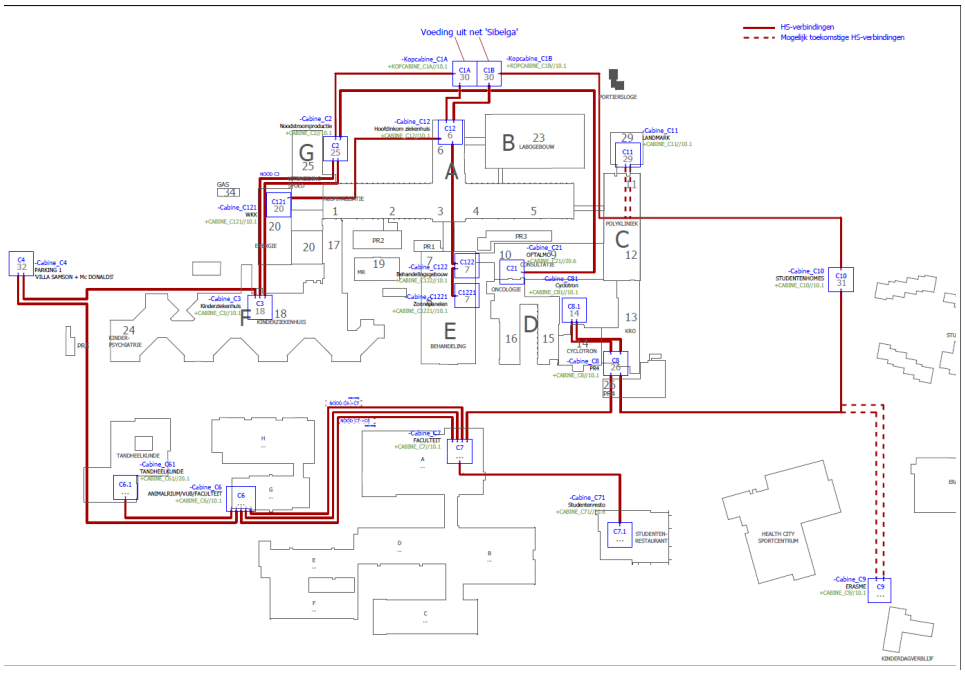
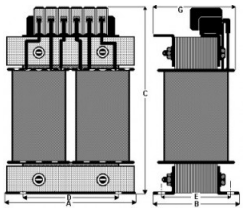


Figure 4.8: UZB electricity distribution network layout

Context Introduction As part of this project the BHC, containing the academic hospital, is a well-advanced energy island owning and running a state-of-the-art micro-grid that can work in island mode for five consecutive days. It includes a thermal and electricity grid, wastewater recovery, a high- speed glass-fibre telecom network and a total of 33 *High Voltage* (HV) transformers divided over 18 HV substations. Energy production and storage includes solar PV, *Combined Heat and Power* (CHP), three emergency diesel generators, and a total capacity of 2,5 MWh in battery storage, mainly under the form of UPS. The micro-grid serves the hospital complex, 250 student dwellings, the faculty of health sciences, a primary school and a fitness centre.



The micro-grid system is conceived to go in island mode with complete automatic transition in maximum 15 seconds in case of critical need and in three minute to comfort need. Cutting edge control technology and maximal reliability are the focus points of this demonstration site. The hospital, our primary focus, has its own distribution network, as shown in Figure 4.8. The topology of the network presents a closed-ring shape for increased reliability and is connected to the grid through two links to nodes C1A and C1B, located at the same place. Each "node" of the network is HV cabin identified by a code {C1, C2, . . . , C12}, with a main transformer. Furthermore it can have connection with one or more "sub" transformers, like the one showed in before, who in turn are connected to "consumers" or power sources. These can vary from an individual room to a whole medical department.

4.2.2 Goal(s), purpose & critical factors

Long term

- ◎ Grow Zensor into the main data hub for the Green Energy Park.
- ◎ Minimizing the energy losses and overall consumption, leading to a more profitable operation.
- ◎ Identify where the exact sources of this cost are and where the best opportunities for improvement lie.

Short term

- ◎ Having a view on the data, centralized and well accessible for multiple people in a structured way.
- ◎ Monitoring and tracking energy consumption in a production site resulting in more than mere energy bill reductions.
- ◎ Improve sustainability reducing energy need and peek request.

4.2.3 Project description by phases

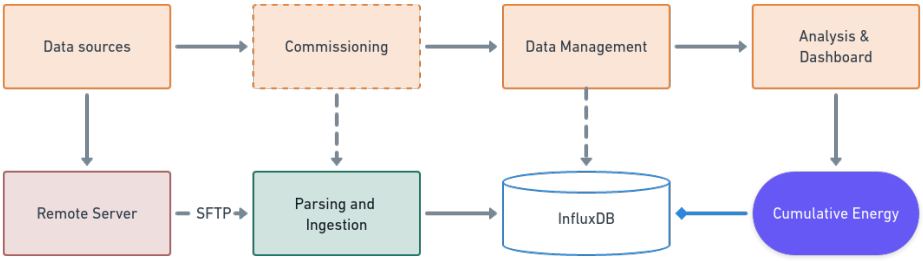


Figure 4.9: VUB's (light) project core stages

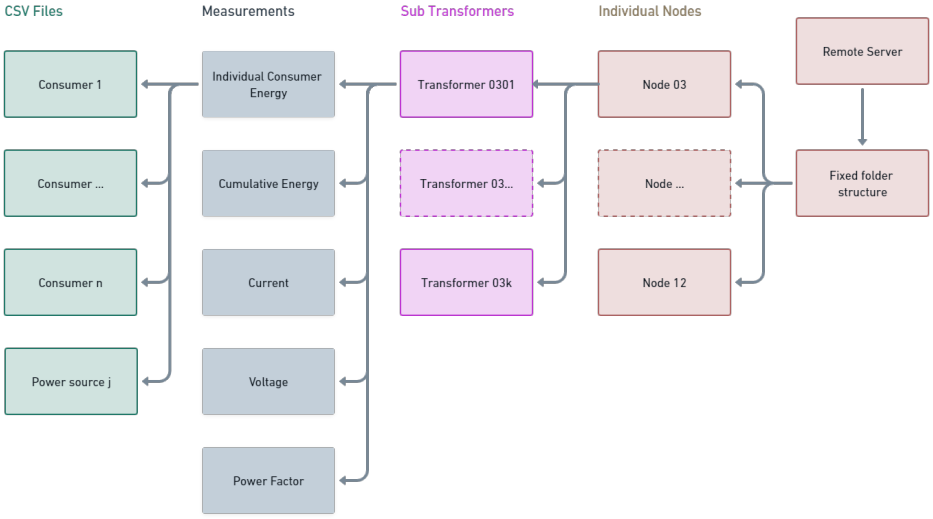


Figure 4.10: VUB's remote server folder tree structure

Datasources MOBI, VUB research group, has collected through the years several MB of 15-minute data energy related out of the distribution network. This limited dataset is currently stored on a remote server in a simple, yet effective way. a folder tree, that tries to reflect reality, is our main metadata's source. Looking at Figure 4.10 from right to left, in a bottom up manner, should help us clarify the situation. Starting from the leaves, we found the "CSV Files". Each file share the same common structure with two columns, *time* and *value*, and multiple rows chronologically sorted. It can either contain information about a secondary transformer or a consumer/power-source as mentioned before in Subsection 4.2.1. To make such distinction we have to rely on his filename, indicating the hardware source and/or the

physical element measured. For example we could find the `Transfo-I3.csv`, referring to transformer's 3° phase current or `Bord LG 03 Radiologie.csv` radiology department energy consumption. How are the different physical quantities organized? To answer this question we go up one level in the tree, moving to the right, stepping up to "Measurements". Here we have five separated directory. These represent different electricity measurement $\{current, voltage, power\,factor, energy\}$, taken on the secondary *Low Voltage* (LV) terminals of the transformers. About the energy (kWh): we have to make an important distinction between individual or cumulative. It can either represent the consumption of one individual consumer or the whole sub transformer (ideally it should be the sum of all consumer connected to it). Instead, for the others (current, voltage e power factor data later added to the dataset), is only available at the sub transformer level, not consumer. Let's now change the way we traverse the tree, from bottom-up to top down. From the root ("Fixed folder structure") we go down to the "Individual Nodes" layer, which is pretty self-explanatory. We find in fact a directory for each node of our distribution network. This folder contains one or more sub-folder, one for each sub-transformer connected to the same node. Going down even further to the "Sub transformer" level, same logic applies. For each sub transformer we have it's one metrics folders. So here it is the connection point. To clarify, let's take the previous example and extend it further: the `Bord LG 03 Radiologie.csv` will have the following path *root/NodeC03/Transfo0302/ConsumerEnergy/Bord...*.

Figure 4.11: VUB's data ingestion flowchart

Commissioning & Data Management Once the university granted the credential to remotely access this dataset, we started managing it, periodically accessing it using *SSH File Transfer Protocol* (SFTP) over port 9921. The **first step** of the ingestion is, as illustrated in the figure 4.11, copying data from VUB's server to Zensor's one. This happens periodically as cron job, see Section 2.3, also thanks to Python library `pysftp`. We recursively traverses the sftp folder and if we comes across a .csv file we copies it into the same folder structure as on the server.

Subsequently, for **step two**, we close the connection and work with our freshly copied data. We traverse, once again, the folder tree, keeping track of folder names that are gonna be important as metadata. Once we are at leaf level, see once again Figure 4.10, we use Pandas (3.1) for file reading, inferring the datetime strings format. This switch to a faster parsing method

can increase the speed by 5-10x. Then we can perform some data cleaning, for eliminating outliers, duplicates and NaNs. Immediately afterwards we tag the respective **DataFrame** with the necessary information to easily identify it, like description, unit and the metadata previously collected.

Since the numerous files are of significant size, this series of operations can take a long time. Therefore, it was decided to parallelize the whole operation using a pool of processes over threads. To substantiate this point, here is a small digression: in Python, if code is CPU-bound, multithreading won't help, because only one thread can hold the Global Interpreter Lock, and therefore run Python code, at a time. So, in this specific scenario we need to use processes, not threads. Indeed in multiprocessing, any newly created process will run independently with its own memory space.

Finally, the **step three** involves writing each DataFrame on the InfluxDB (3.2), our choice for Time Series Database. This operation, which is already quite optimized, can be easily executed in parallel given the limited concurrency. As a result we will have a "measurement" for each node, which will contain several series, each one easily distinguishable from the others.

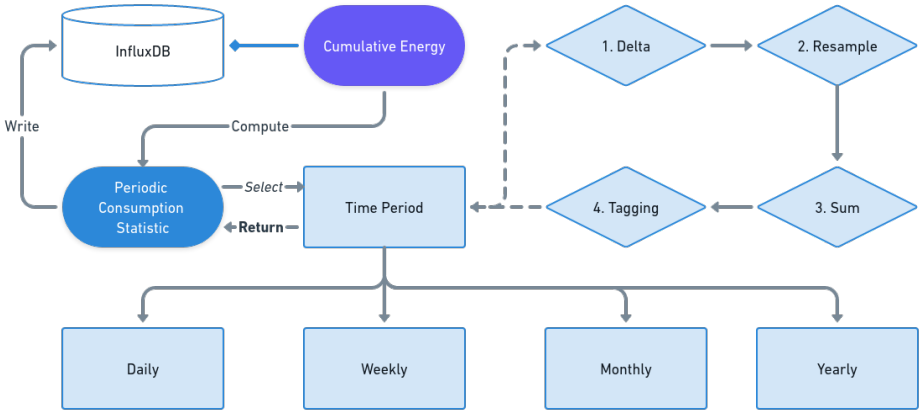


Figure 4.12: VUB's analytics chart

Analysis They want to see some basic visualizations of this limited dataset, no live link or permanent ingestion will have to be set up. In a first instance the dashboards need to have the same structure as the folders the .csv files are in. So: each folder containing .csv files should be a page in Grafana displaying the graphs of the series inn the .csv's in that folder. In a second instance they want the entry point to be a map of the campus with the main transformers indicated and with a clickable link on each item such that the data can be consulted.

4.2.4 Conclusion

Appendix A

Acronyms

Infralytics© Infrastructure Analytics, 11

AI Artificial Intelligence, 8

AWS Amazon Web Services, 18

BHC Brussels Health Campus, 44

BI Business Intelligence, 1

CBM Condition-based maintenance, 8

CDA Confirmatory data analysis, 1

CHP Combined Heat and Power, 44

CM Condition monitoring, 9

CMMS Computerized Maintenance Management Systems, 8

CRISP Cross-industry standard process, 2

CRUD Create Read Update Delete, 28

CSV Comma-separated values, 21

CtM Continuos monitoring, 35

DM Data mining, 1

EDA Exploratory data analysis, 1

GEP Green Energy Park, 44

HV High Voltage, 44

IoT Internet Of Things, 8

IPCs Industrial Personal Computers, 15

KPIs Key Performance Indicators, 4

LV Low Voltage, 46

ML Machine Learning, 9

MRO Maintenance, repair and overhaul, 6

PdM Predictive maintenance, 8

PLC Programmable Logic Controller, 43

PM Preventive maintenance, 7

PPM Planned preventive maintenance, 7

RMS Root mean square, 40

S3 Simple Storage Service, 18

SaaS Service as a Service, 15

SFTP SSH File Transfer Protocol, 47

TSDB Time Series Database, 25

UZB Universitair Ziekenhuis Brussel, 44

VUB Vrije Universiteit Brussel, 44

List of Tables

1.1 NOIR system of classification of types of data (Source: [4]) . . 4

2.1 Assets, Industries, and Infrastructure for which Zensor has specific products 14
2.2 1° and 2° Phases differences of Zensor's Approach 16

3.1 Sample time series dataset: number of butterflies and honeybees counted by two scientists 27
3.2 Pros and cons of InfluxDB 28
3.3 Time units and relative ranges example 31

4.1 Stumabo's log file structure 36
4.2 intermediate result: numerical RMS `1Hz` values are given using engineering notation: 10^{-6} 43

List of Figures

1.1 Data science process flowchart (Source: [3]) 2
1.2 *USS Ronald Reagan (CVN 76) Undergoes Preventive Maintenance* (Source: [11]) 7
1.3 Condition-based maintenance schema (Source: [15]) 9
1.4 Service as a Service Workflow (Source: [18]) 11

2.1 Building blocks for full project 15
2.2 Building blocks for light project 16
2.3 Zensor's system architecture for data analytics 18
2.4 Analysis script structure 20

3.1 DataFrame: Pandas' core data structure 22
3.2 Shows the Split-Apply-Combine using an aggregation function (Source: [26]) 23
3.3 Examples of signal waveform data processed by resample . . . 24
3.4 Random data plus trend, with best-fit line and different smoothing applied. (source: Wikimedia Commons [27]) 26
3.5 Official data and insights on coronavirus (Source: gov.uk) . . 29
3.6 Grafana Graph Visualization (Source: Flickr [39]) 30

4.1 Stumabo's blade production line covered by this project . . . 34
4.2 Stumabo's (full) project core stages 35
4.3 Station 1 blue: sensor setup 36
4.4 Stumabo's engineering schematic, also relevant during analysis 38
4.5 Exploratory data analysis on 1° block, station 1 blue 39
4.7 RMS amplitude comparison between low and high frequency data 42
4.8 UZB electricity distribution network layout 44
4.9 VUB's (light) project core stages 46
4.10 VUB's remote server folder tree structure 46
4.11 VUB's data ingestion flowchart 47
4.12 VUB's analytics chart 48

Bibliography

[1] Meta S. Brown. "Transforming Unstructured Data into Useful Information". In: Big Data, Mining, and Analytics. Auerbach Publications, Mar. 2014, pp. 227–246. DOI: 10.1201/b16666. (Visited on 08/26/2021).

[2] Claude A. Pruneau. "The Multiple Facets of Correlation Functions". In: *Data Analysis Techniques for Physical Scientists*. Cambridge University Press, 2017, pp. 526–576. DOI: 10.1017/9781108241922.013.

[3] Schutt Rachel and O'Neil Cathy. *Doing data science*. O'Reilly Media, 2013, pp. 1–30, 120–140. ISBN: 9781449358655.

[4] Vicki Adams. "Introduction to data analysis". In: *The Journal of small animal practice* 49 (Sept. 2008), pp. 375–6. DOI: 10.1111/j.1748-5827.2008.00647.x.

[5] James Goodnight. "The forecast for predictive analytics: hot and getting hotter". In: *Statistical Analysis and Data Mining* 4 (Jan. 2011), pp. 9–10. DOI: 10.1002/sam.10106. (Visited on 06/16/2019).

[6] John W. Tukey. "The Future of Data Analysis". In: *The Annals of Mathematical Statistics* 33.1 (1962), pp. 1–67. DOI: 10.1214/aoms/1177704711. URL: https://doi.org/10.1214/aoms/1177704711.

[7] Wikipedia Contributors. *Data Cleansing*. Wikipedia, Apr. 2019. URL: https://en.wikipedia.org/wiki/Data_cleansing (visited on 03/10/2022).

[8] Charles M. Judd. *Data analysis: a model-comparison approach*. Harcourt Brace Jovanovich, 1989, pp. 10–15. URL: https://openlibrary.org/books/OL18760366M/Data_analysis (visited on 03/14/2022).

[9] European Federation of Maintenance Societies. *EFNMS: What Does EFNMS Stand For?* http://www.efnms.eu/, 2016. URL: http://www.efnms.eu/about-us/what-does-efnms-stand-for/ (visited on 03/07/2022).

[10] *Paints and varnishes – Corrosion protection of steel structures by protective paint systems – Part 9: Protective paint systems and laboratory performance test methods for offshore and related structures*. Tech. rep. Geneva, CH: International Organization for Standardization, Jan.

2018. URL: https://www.iso.org/standard/64832.html (visited on 03/07/2022).

[11] U.S. Indo-Pacific Command. *USS Ronald Reagan (CVN 76) Undergoes Preventive Maintenance*. flickr, Jan. 2016. URL: https://www.flickr.com/photos/us-pacific-command/23569479804/ (visited on 03/07/2022).

[12] Michael Decourcy Hinds. "PREVENTIVE MAINTENANCE: A CHECK LIST". In: *The New York Times* (Feb. 1985). URL: https://www.nytimes.com/1985/02/17/realestate/preventive-maintenance-a-checklist.html (visited on 02/28/2022).

[13] Jonathan Trout. *Preventive Maintenance: An Overview*. Reliableplant, Sept. 2019. URL: https://www.reliableplant.com/Read/12494/preventive-maintenance (visited on 02/28/2022).

[14] Daniel Penn. *What is Preventive & Predictive Maintenance?* Daniel Penn Associates, Jan. 2020. URL: https://www.danielpenn.com/preventive-predictive-maintenance-2020/ (visited on 03/08/2022).

[15] Bizerba S.p.A. *Condition-based maintenance*. Bizerba.com. URL: https://www.bizerba.com/it_it/argomenti/digital-services/ (visited on 03/09/2022).

[16] Leith Hitchcock. "ISO Standards for Condition Monitoring". In: Engineering Asset Management. Springer, 2012, pp. 606–613. DOI: 10.1007/978-1-84628-814-2_65. (Visited on 03/08/2022).

[17] Valerio Dida et al. *Manufacturing: Analytics unleashes productivity and profitability | McKinsey*. Mckinsey, 2020. URL: https://tinyurl.com/mckinseybusiness.

[18] Yves Van Ingelgem and Marteen Durie. *Zensor: our approach*. Zensor, 2022. URL: https://www.zensor.be/our-approach (visited on 01/16/2022).

[19] Yves Van Ingelgem and Marteen Durie. *What Does The Future Of Maintenance Look Like?* Zensor, Dec. 2020. URL: https://www.zensor.be/blog/futureofmaintenance (visited on 03/07/2022).

[20] Matthew Le Merle. "It's Time for Service as a Service". In: *Financial Times* (Oct. 2012). URL: https://www.ft.com/content/f88bc87a-0e4b-11e2-8b92-00144feabdc0 (visited on 03/16/2022).

[21] "Chip shortage: Toyota to cut global production by 40%". In: *BBC News* (Aug. 2021). URL: https://www.bbc.com/news/business-58266794.

[22] *License:OLDAP-2.7 – Free Software Directory*. 1999. URL: https://directory.fsf.org/wiki/License:OLDAP-2.7 (visited on 01/31/2022).

[23] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[24] *IO tools (text, CSV, HDF5, ...) – pandas 1.4.0 documentation*. URL: https://pandas.pydata.org/docs/user_guide/io.html (visited on 01/31/2022).

[25] *pandas-dev/pandas: Pandas 1.4.0*. Jan. 2022. DOI: 10.5281/zenodo.5893288. URL: https://doi.org/10.5281/zenodo.5893288.

[26] Anurag Pandey. *Split-Apply-Combine Strategy for Data Mining*. en. Oct. 2020. URL: https://medium.com/analytics-vidhya/split-apply-combine-strategy-for-data-mining-4fd6e2a0cc99 (visited on 02/28/2022).

[27] *File:Random-data-plus-trend-r2.png - Wikimedia Commons*. en. URL: https://commons.wikimedia.org/wiki/File:Random-data-plus-trend-r2.png (visited on 02/08/2022).

[28] Matt Asay. *Why time series databases are exploding in popularity*. en-US. June 2019. URL: https://www.techrepublic.com/article/why-time-series-databases-are-exploding-in-popularity/ (visited on 02/08/2022).

[29] Michael Duffy. *DevOps Automation Cookbook*. en. Google-Books-ID: k_SoCwAAQBAJ. Packt Publishing, Nov. 2015. ISBN: 9781784398392.

[30] *InfluxDB: Open Source Time Series Database*. URL: https://www.influxdata.com/ (visited on 02/06/2022).

[31] The Graphite Project. *Graphite Documentation, v1.1.8*. Graphite, Apr. 2021. URL: https://graphite.readthedocs.io/en/1.1.8 (visited on 03/09/2022).

[32] *InfluxDB OSS 1.8 Documentation*. URL: https://docs.influxdata.com/influxdb/v1.8/ (visited on 01/31/2022).

[33] Syeda Noor et al. *Université libre de Bruxelles Advanced Databases Time Series Databases and InfluxDB*. 2017. URL: https://cs.ulb.ac.be/public/_media/teaching/influxdb_2017.pdf (visited on 02/18/2022).

[34] *Grafana: The open observability platform*. en. URL: https://grafana.com/ (visited on 02/06/2022).

[35] *Grafana documentation*. en. URL: https://grafana.com/docs/grafana/latest/ (visited on 01/31/2022).

[36] W3Techs. *Usage Statistics and Market Share of Google Analytics for Websites*. en. W3techs. URL: https://w3techs.com/technologies/details/ta-googleanalytics (visited on 02/28/2022).

[37] *UK Summary | Coronavirus (COVID-19) in the UK*. en. URL: https://coronavirus.data.gov.uk (visited on 02/14/2022).

[38] *GLAM Wiki Dashboard*. URL: https://glamwikidashboard.org/about (visited on 02/14/2022).

[39] Linux Screenshots. *grafana dashboard*. Jan. 2016. URL: https://www.flickr.com/photos/xmodulo/24311604930/ (visited on 02/14/2022).

[40] Sunil Kumar and Prof. Saravanan. "A Comprehensive study on Data Visualization tool - Grafana". English. In: 8.5 (May 2021), pp. 1–7. ISSN: ISSN-2349-5162. URL: https://www.jetir.org/papers/JETIR2105788.pdf.

[41] @2022 Stumabo. *Stumabo*. 2022. URL: https://www.stumabo.com/en (visited on 01/23/2022).

[42] VUB. *Green Energy Park | Green Energy Park*. Green Energy Park, Jan. 2020. URL: https://www.greenenergypark.be/green-energy-park-2/?lang=en (visited on 03/25/2022).