FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS ENGINEERING DEPARTMENT
UNDERGRADUATE COURSE IN CONTROL AND AUTOMATION ENGINEERING

Fernando Rech Piccoli

Application of Data Analysis Algorithms for Cost, Time, and Quality Improvement in Machining Processes

Aachen
2025

Fernando Rech Piccoli

Application of Data Analysis Algorithms for Cost, Time, and Quality Improvement in Machining Processes

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergraduate Course in Control and Automation Engineering of the Federal University of Santa Catarina.
Supervisor: Prof. Carlos Barros Montez, Dr.

Aachen
2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Rech Piccoli, Fernando
Application of Data Analysis Algorithms for Cost, Time, and Quality Improvement in Machining Processes / Fernando Rech Piccoli ; orientador, Carlos Barros Montez, 2026.
98 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2026.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Data Analysis. 3. Machining. 4. Improvement. 5. Algorithms. I. Barros Montez, Carlos. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Fernando Rech Piccoli

Application of Data Analysis Algorithms for Cost, Time, and Quality Improvement in Machining Processes

This dissertation was evaluated in the context of the subject DAS5511 (Course Final Project) and approved in its final form by the Undergraduate Course in Control and Automation Engineering

Florianópolis, March 4-th, 2026.

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Course Coordinator

Examining Board:

Prof. Carlos Barros Montez, Dr.
Advisor
UFSC/CTC/EAS

ICP-Edu
Documento assinado digitalmente
Carlos Barros Montez
Data: 27/03/2026 15:12:00-0300
CPF: ***.035.027-**
Verifique as assinaturas em https://v.ufsc.br

Tommy Venek, M.Sc.
Supervisor
gemineers GmbH

T. Venek

Prof. Rodrigo Lange, Dr.
Evaluator
IFSC

Prof. Eduardo Camponogara, Dr.
Board President
UFSC/CTC/EAS

This work is dedicated to my classmates and my dear parents and sister.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to all those who supported me throughout the development of this project.

First, my sincere thanks to my friends at the company - Maurício, Arthur, Pedro, Rhanna, Lucas, Giovanna, Paulo, Fernando, Gustavo, Pietro, Victor -, for their friendship and support and to Ícaro, for his help troughout this project.

To my friends from "Covil" that I made in Aachen, who made this journey easier and fun.

I would also like to thank my friend group "Arca do Caos" from the university, who accompanied me along the graduation and with whom I created great memories.

To my family, especially my parents - Alexandre and Silvia - and my sister - Georgia - for their unwavering love, support and encouragement in everything that I do - and to my girlfriend - Gabriela - for her patience, love and motivation throughout this journey.

Finally, I am deeply thankful to Tommy and Vincent for providing me with this incredible opportunity and experience.

“Primary causes are unknown to us; but are subject to simple and constant laws, which may be discovered by observation, the study of them being the object of natural philosophy." (FOURIER, 1822)

DISCLAIMER

Aachen, March 4-th, 2026

As representative of the gemineers GmbH in which the present work was carried out, I declare this document to be exempt from any confidential or sensitive content regarding intellectual property, that may keep it from being published by the Federal University of Santa Catarina (UFSC) to the general public, including its online availability in the Institutional Repository of the University Library (BU). Furthermore, I attest knowledge of the obligation by the author, as a student of UFSC, to deposit this document in the said Institutional Repository, for being it a Final Program Dissertation (*"Trabalho de Conclusão de Curso"*), in accordance with the *Resolução Normativa n° 126/2019/CUn*.

T. Venek

Tommy Venek
gemineers GmbH

ABSTRACT

This work presents the development and evaluation of an integrated data-driven framework for the analysis, visualization, and interpretation of machining process data, aiming to support decision-making related to time efficiency, tooling costs, and product quality. The research adopts a methodological approach based on the systematic processing of operational data acquisiton obtained from machining operations, combining interactive time-series visualization, frequency-domain inspection, and performance-oriented analytical modules. The framework enables structured selection of operations, comparison of raw and processed signals, and exploratory analysis across multiple temporal and operational contexts. Based on these analyses, optimization scenarios are evaluated, including alternative tool selection strategies, identification of inefficient operating conditions, and early recognition of potentially harmful process behavior. The results indicate that meaningful reductions in machining time and tooling costs could be achieved by improving tool utilization, reducing unnecessary air-cutting and low-efficiency movements, and mitigating abnormal load or force conditions before tool or part failure occurs. Additionally, qualitative improvements in process stability and product quality are inferred through the reduction of unfavorable operating conditions. Although the proposed framework is not validated under real-time industrial deployment, the results demonstrate its potential as a practical decision-support tool for manufacturing environments, highlighting the value of integrated signal analysis and interactive visualization in supporting process optimization and continuous improvement initiatives.

Keywords: Data-driven framework. Machining process data. Cost time and quality. Analytical modules. Integrated signal analysis. Interactive visualization. Process Optimization.

RESUMO

Este trabalho apresenta o desenvolvimento e a avaliação de uma estrutura integrada baseada em dados para análise, visualização e interpretação de informações de processos de usinagem, com o objetivo de apoiar a tomada de decisão relacionada à eficiência de tempo, aos custos de ferramentais e à qualidade do produto. A pesquisa adota uma abordagem metodológica fundamentada no processamento sistemático de dados operacionais provenientes de operações de manufatura, combinando visualização interativa de séries temporais, inspeção no domínio da frequência e módulos analíticos orientados ao desempenho do processo. A estrutura proposta permite a seleção estruturada de operações, a comparação entre sinais brutos e processados e a análise exploratória em diferentes contextos temporais e operacionais. A partir dessas análises, são avaliados cenários de otimização, incluindo estratégias alternativas de seleção de ferramentas, identificação de condições operacionais ineficientes e detecção antecipada de comportamentos potencialmente prejudiciais ao processo. Os resultados indicam que reduções significativas no tempo de usinagem e nos custos associados às ferramentas podem ser obtidas por meio da melhoria do uso das ferramentas, da redução de movimentos improdutivos e da mitigação de condições anômalas de carga e esforço antes da ocorrência de falhas da ferramenta ou da peça. Além disso, infere-se uma melhoria qualitativa na estabilidade do processo e na qualidade do produto. Embora o sistema não tenha sido validado em ambiente industrial em tempo real, os resultados demonstram seu potencial como uma ferramenta de apoio à decisão em contextos de manufatura, reforçando a relevância da análise integrada de sinais e da visualização interativa de dados para iniciativas de otimização e melhoria contínua.

Palavras-chave: Estrutura baseada em dados. Dados de processos de usinagem. Tempo, custo e qualidade do processo. Módulos analíticos. Análise integrada de sinais. Visualização interativa de dados. Otimização de processos.

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
CNC	Computer Numerical Control
DAQ	Data Acquisition
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
HBW	Brinell hardness (tungsten carbide indenter)
HSS	High-Speed Steel
IPW	In-Process Workpiece
JSON	JavaScript Object Notation
KDE	Kernel Density Estimation
KPI	Key Performance Indicator
MAD	Median Absolute Deviation
NoSQL	Non-relational / non-SQL database
NTSB	National Transportation Safety Board
PSD	Power Spectral Density
REST	Representational State Transfer
RPM	Revolutions per minute
SG	Savitzky–Golay
STFT	Short-Time Fourier Transform
STL	Standard Tessellation Language
ZIP	Compressed archive format

LIST OF SYMBOLS

t_{total}	Total operation time
t_{engaged}	Engaged time (effective cutting time)
r_{XY}	Pearson correlation coefficient between X and Y
X_i	i-th observation of variable X
Y_i	i-th observation of variable Y
$\bar{X}$	Sample mean of X
$\bar{Y}$	Sample mean of Y
n	Number of observations (sample size)
X_i^{norm}	Normalized (min–max) value of X_i
X_{min}	Minimum value of X
X_{max}	Maximum value of X
f_c	Filter cutoff frequency
N	Window size (number of samples) in filters/averages
x_i	i-th sample of the original signal
y_i	i-th sample of the filtered signal (e.g.
$G(x)$	Gaussian function (Gaussian filter kernel)
μ	Mean (center) of the Gaussian distribution
σ	Standard deviation (width) of the Gaussian
$(f * G)(x)$	Convolution of f with the Gaussian kernel
$\hat{f}(x)$	Smoothed estimate (e.g.
h	Bandwidth of KDE / smoothing parameter
$K(\cdot)$	Kernel function used in KDE
M	Half-window size of the Savitzky–Golay filter (window $2M + 1$)
p	Degree of the local polynomial in the Savitzky–Golay filter
a_j	Coefficient j of the local polynomial in Savitzky–Golay
$\hat{s}[n_0]$	Smoothed value at the window center (Savitzky–Golay)
h_k	Filter coefficients (convolution) in Savitzky–Golay
f_s	Spindle rotation frequency
N_s	Spindle speed (in RPM)
f_t	Tooth passing frequency
Z	Number of teeth/cutting edges of the tool
P_i^{raw}	Raw (non-normalized) priority of operation i
P_i	Normalized priority of operation i
T_i	Total time of operation i
E_i	Engagement (fraction/percentage) of operation i
ε	Small constant to avoid division by zero
$X(f)$	Fourier transform of $x(t)$

j	Imaginary unit
$X[k]$	Discrete Fourier Transform (DFT) at index k

CONTENTS

1 INTRODUCTION

This dissertation, entitled *Application of Data Analysis Algorithms for Cost, Time, and Quality Improvement in Machining Processes*, explores the development and implementation of data-driven methodologies to enhance efficiency in modern manufacturing. By integrating advanced algorithms for data analysis into machining workflows, the study aims to provide systematic strategies that reduce production costs, optimize processing time, and ensure consistent quality standards. The research is carried out in collaboration with gemineers GmbH, a company specialized in developing digital twins for manufacturing processes in the automotive and aerospace sectors. The work combines theoretical foundations with applied solutions, bridging the gap between academic research and industrial application.

1.1 CONTEXT AND MOTIVATION

The increasing complexity of manufacturing processes in high-precision industries such as automotive and aerospace demands innovative solutions that go beyond traditional trial-and-error methods. Companies face growing pressure to shorten development cycles, minimize resource consumption, and maintain high product quality, all while operating in a competitive and sustainability-driven market. Digital twins and data analysis algorithms present a transformative opportunity, enabling virtual testing, predictive optimization, and real-time monitoring of machining processes. Within this context, the motivation for this dissertation lies in leveraging data science techniques not only to address existing inefficiencies in machining operations but also to establish a scalable framework for continuous improvement. By applying advanced analytical methods, the goal is to create tangible industrial impact—reducing costs, saving time, and ensuring robust quality control.

1.2 ABOUT THE COMPANY: GEMINEERS GMBH

gemineers GmbH is a German deep-tech startup that emerged from Fraunhofer IPT in July 2021 (Fraunhofer IPT, 2026b). The company is based in Aachen and its primary focus is to accelerate the digitalization of machining processes through digital twins.

Figure 1 – gemineers GmbH logo

Source: Company‘s archive.

The gemineers product offers a comprehensive software solution that harnesses digital twin technology to monitor machining processes. By directly collecting data from machine tools, the software achieves complete digitalization of machining products and processes, enabling easier quality assessment and product evaluation. Gemineers' platform for data-driven quality assurance not only provides access to previously untapped manufacturing data but also facilitates digital assessment of component quality. Users benefit from thorough digital documentation of their processes and products, reducing both time and financial investments typically associated with physical quality assurance and product evaluation. Further details about the software will be discussed in section 2.2.

1.3 PROBLEM OVERVIEW AND IMPORTANCE

Despite the growing adoption of digital twins and data-driven tools in manufacturing, the full potential of these technologies is often underutilized in machining processes. Traditional approaches to process optimization frequently rely on operator expertise, empirical adjustments, or post-process inspections, which can lead to inefficiencies, increased costs, and variability in product quality. In industries such as automotive and aerospace—where tolerance levels are extremely tight and production volumes are high—even small deviations in machining parameters can result in significant material waste, rework, or tool wear, directly impacting profitability and delivery timelines.

Another critical challenge lies in the sheer volume and complexity of data generated during machining operations. Parameters such as spindle speed, feed rate, tool wear, vibration, and cutting forces interact in nonlinear ways that are not easily captured by conventional statistical methods. Without advanced analysis techniques, valuable insights remain hidden, preventing companies from identifying root causes of inefficiencies or predicting process anomalies before they occur.

Addressing this problem is of strategic importance for manufacturers aiming to remain competitive in a global market that prioritizes efficiency, sustainability, and quality assurance. By applying modern data analysis algorithms within the digital twin framework, it becomes possible to move from reactive problem-solving toward predictive and prescriptive decision-making. This not only improves machining performance but also supports broader organizational goals, such as reducing costs, minimizing downtime, ensuring high product quality, and accelerating innovation cycles.

1.4 OBJECTIVES

The general objective of this dissertation is to develop a comprehensive framework capable of delivering automated and intuitive insights into machining milling process performance with respect to cost, time, and quality. This framework leverages algorithms and statistical methods to transform raw process data into actionable knowledge, supporting decision-making and enabling continuous improvement within manufacturing operations.

The specific objectives of this work are structured around the main dimensions of machining performance:

- Sorting of operations based on total operation time and engaged cutting time to highlight inefficiencies.
- Time analysis including feedrate and spindle speed evaluation to identify potential optimization opportunities.
- Cost analysis focused on tool usage and wear, linking operational strategies to economic impact.
- Quality analysis divided into load monitoring (stress and tool failure risks) and frequency evaluation (process stability and vibrations).
- Additional time series visualization module for flexible and detailed exploration of machining data.
- Integration of all analyses into a Streamlit-based frontend, ensuring an intuitive and interactive interface.

By addressing these objectives, the framework seeks to not only detect inefficiencies and anomalies in machining processes but also to provide actionable recommendations for improvement. The integration of these analyses into a cohesive system enhances the transparency and usability of digital twins, supporting both engineers and decision-makers in achieving measurable improvements in cost, time, and quality.

1.5 PROPOSED SOLUTION OVERVIEW

To achieve the general objective, the project is divided into a set of components, each focusing on distinct aspects of process analysis and optimization:

(a) **Ordering by Priority**

- The framework implements an ordering system that ranks machining operations based on their relative contribution to overall process time and engagement time.
- This prioritization allows engineers to quickly identify the most time-consuming or inefficient operations, enabling targeted optimization efforts.
- The ranking metric combines both the total operation duration (t_{total}) and the actual engagement duration (t_{engaged}), reflecting not only how long an operation takes, but also how effectively that time is utilized for material removal.
- A higher priority value indicates operations that occupy a significant portion of the total cycle time but exhibit a low engagement ratio, representing potential for time reduction or process improvement.

(b) **Cost and Time Analysis**

- Evaluate key performance indicators (KPIs) such as engagement time (e.g., air cuts, idle time, wasted time).
- Assess tool depth and angle usage to better understand process efficiency.
- Estimate potential time savings through feedrate analysis and optimization.

(c) **Load and Force Analysis**

- Identify high-load points during machining and provide early warnings of potential tool breakage.
- Map these events directly onto a 3D representation of the tool path, enhancing interpretability and root-cause analysis.

(d) **Frequency Analysis**

- Apply frequency-domain techniques such as Fast Fourier Transform (FFT), Short-Time Fourier Transform (STFT), and frequency filtering.
- Correlate frequency responses with the tool path in 3D, facilitating the detection of chatter, vibration issues, and other dynamic instabilities.

(e) **Time Series Visualization**

- Develop a customizable interface for the visualization of process time series data.
- Allow users to tailor the analysis to specific needs, enabling deeper insights and improved decision-making flexibility.

(f) **Streamlit-Based Frontend**

- Implement a web-based application using Streamlit to integrate all analysis modules.
- Provide an intuitive, interactive, and user-friendly interface for engineers and decision-makers.

1.6 METHODOLOGY AND TOOLS USED

The development of the proposed framework required the integration of different methodologies, algorithms, and tools to cover the main aspects of machining process analysis. The work combined data-driven approaches with domain-specific knowledge to ensure that the results were both technically rigorous and practically relevant for manufacturing environments.

The main methodologies and tools applied can be summarized as follows:

- **Data Analysis Algorithms:** Methods for processing and transforming raw machining signals into structured and meaningful data.
- **Statistical Algorithms:** Correlation analysis and related techniques for identifying relationships among machining parameters.
- **Fourier Transformations:** Application of FFT and STFT for frequency-domain analysis and vibration detection.
- **Gaussian Filters:** Signal smoothing and noise reduction to improve data quality before analysis.
- **Frontend Interface:** Development of an interactive and intuitive visualization platform using Streamlit.
- **Backend API Management:** Data handling, communication between modules, and system integration.
- **Milling Process Knowledge:** Fundamental understanding of machining operations, tool behavior, and process dynamics to interpret and validate results.

1.7 RESULTS AND IMPACT

Overall, the results indicate that the proposed framework can meaningfully improve machining performance across quality, cost, and time dimensions. The analyses showed that abnormal cutting behavior, tool degradation, and unstable conditions can be identified early or help find the problem, if it occurs, using process signals, enabling intervention before tool breakage or part rejection occurs. This capability implies improved process stability, better surface quality, and reduced scrap and rework. From a cost perspective, the results highlight clear opportunities for optimizing tool selection and usage, such as avoiding over-dimensioned tools, improving cutting angle utilization, and preventing premature tool changes, all of which can lead to tangible reductions in tooling and downtime costs. In terms of productivity, the identification of inefficient feedrate usage, excessive air cutting, and non-optimal motion patterns reveals significant potential for machining time reduction. Even modest time savings per operation accumulate across batch production, increasing machine availability and reducing cost per part. These results, grounded in real industrial data, demonstrate that the framework has strong potential to deliver economically relevant benefits when applied in practical manufacturing environments.

1.8 DOCUMENT STRUCTURE

The remainder of this document is organized as follows:

- **Chapter 2 – Theoretical Background:** Provides the scientific and technical foundations necessary to understand the proposed approach, including concepts from data analysis, signal processing, and machining processes.
- **Chapter 3 – Problem Description and Technical Requirements:** Details the industrial context, the company background, and the technical requirements that guide the development of the solution.
- **Chapter 4 – Proposed Solution and Methodology:** Describes the developed framework, the algorithms, and the workflow designed to address the identified challenges.
- **Chapter 5 – Implementation and Results:** Explains the implementation of the solution, the integration of the modules, and the results obtained from applying the framework to machining data.
- **Chapter 6 – Conclusion:** Summarizes the main findings, discusses the impact of the work, and highlights opportunities for future research and development.

2 THEORETICAL BACKGROUND

This chapter encompasses the necessary fundamental concepts and theories for comprehending the work presented in this document.

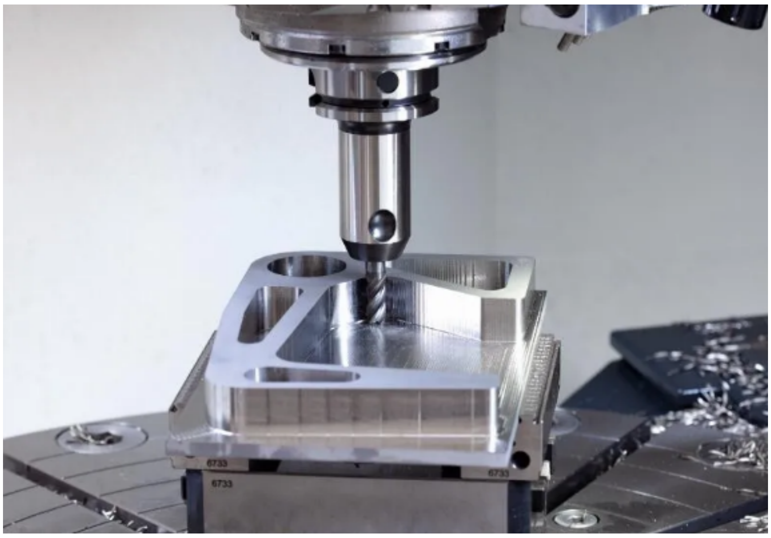
2.1 MACHINING PROCESSES

The machining process uses machine tools to cut, shape, and remove material from a workpiece, transforming it into a desired part, tool, or instrument. It is a subtractive manufacturing method, meaning material is removed rather than added. Common machining processes include turning, where a workpiece rotates against a stationary tool; milling, where a rotating cutter removes material from a stationary or moving workpiece; and drilling, which creates holes. This versatile process is used across industries to create precise components for products ranging from aerospace parts to consumer electronics.

2.1.1 Milling

Milling is a common process that uses rotating, multi-point cutting tools to remove material from a workpiece and create desired shapes, such as flat surfaces, curves, or intricate features. Performed by a milling machine, this process advances the spinning cutter into the stationary or moving workpiece, removing material to achieve high precision and a good surface finish for various parts.This is the type of process that will be approached through the analysis of this report, giving emphasis in the effects of flute length utilization in end mills and angular contact conditions in ball mills, aiming to correlate their impact on tool health, deflection response, and wear evolution during milling operations (Alpakjian; Schmid, 2001).

Figure 2 – Milling Process Example



Source: (CNCLATHING, 2026)

2.1.1.1 End Mill

End mills are versatile cutting tools characterized by their flat cutting ends and multiple flutes along the sides. They are primarily used for profile milling, slotting, contouring, and plunging operations. Unlike drill bits, which cut only in the axial direction, end mills can remove material both axially and radially, allowing them to machine complex geometries with high precision. The tool geometry—such as helix angle, number of flutes, and coating—directly influences cutting performance, chip evacuation, and surface quality. End mills are commonly made of high-speed steel (HSS), carbide, or coated carbide materials to withstand high cutting temperatures and forces.

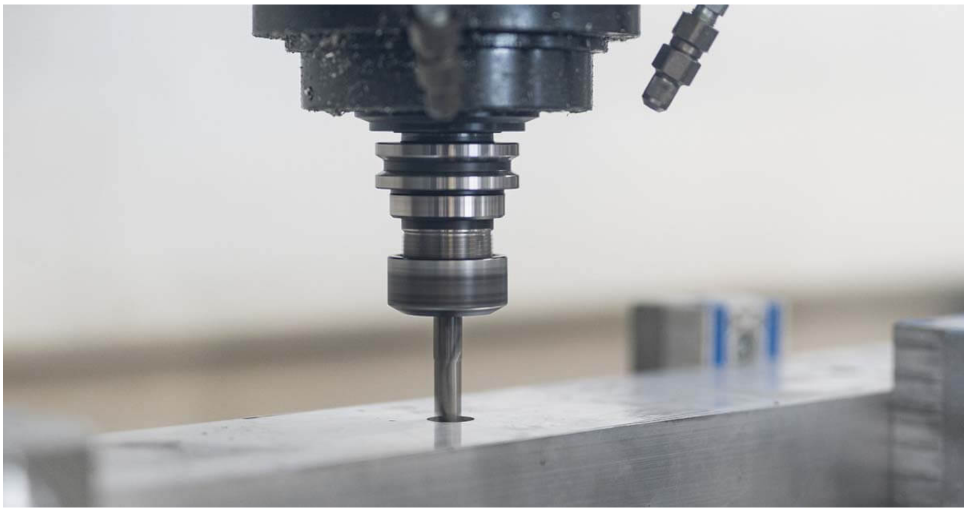
2.1.1.2 Ball Mill

Ball mills, or ball-end mills, feature a hemispherical cutting end that enables smooth contouring and 3D surface machining. They are especially useful for machining complex shapes, such as molds, dies, and curved surfaces, where sharp corners are undesirable or impossible to achieve. The rounded tip reduces stress concentration on the cutting edge and improves surface finish, making the tool ideal for finishing operations.

2.1.2 Drilling

Drilling in machining is a process for creating a hole in a material by using a rotating cutting tool, called a drill bit, which continuously removes layers of material. This operation can be performed using a traditional drill press or a CNC machine, and the primary movements involve the rotation of the tool or workpiece and a linear feed motion to push the tool into the material. The main goal is to create a round hole, though different drilling techniques can also enlarge existing holes, create stepped diameters, or prepare a surface for other components like screws and bolts (Shigley; Mischke, 2015).

Figure 3 – Drilling Process Example



Source: (Ecoreprap, 2026)

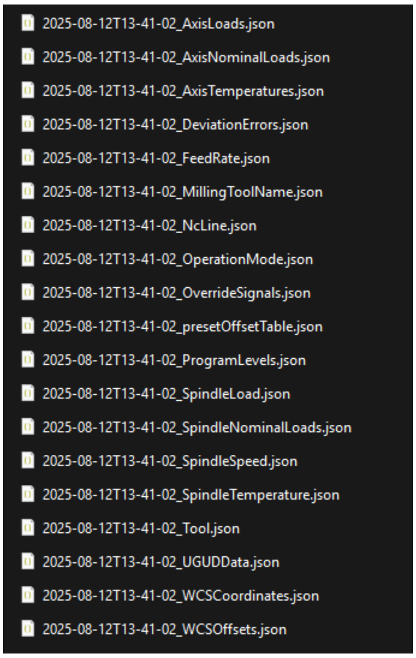
2.2 DIGITAL TWINS SIMULATION AND DATA

2.2.1 Clean and Raw Data

The data collected directly from the machining operations is initially considered raw data, as it is acquired without any processing or integration. This data is captured by the DAQ (Data Acquisition) system and is typically organized into separate files according to the type of information recorded. For example, feedrate measurements are stored in a feedrate.json file, forces in loads.json, spindle speed in another file, and so on. Each file contains a time series of measurements corresponding to the specific signal, which reflects the state of the machine during the operation but has not yet been analyzed or corrected for noise and inconsistencies.

To transform this raw data into a more meaningful and actionable format, it passes through the software framework called "cleaner", which will be deepened in chapter three. During this process, new quantities are calculated from the raw signals, such as chip contact area, resultant forces, and other derived machining parameters that provide insight into process performance. Additionally, the signals undergo pre-processing steps including filtering, synchronization, and aggregation, ensuring that noise is minimized and measurements are aligned across different sensors. Finally, the processed and derived information is integrated into a single, coherent clean data file, called cleanData.json. This consolidated dataset facilitates further analysis, visualization, and interpretation within the framework, providing a reliable foundation for cost, time, and quality assessments of machining operations.

Figure 4 – Raw Data File Example



Source: Author's Archive

2.2.2 Simulation Signals

The framework relies on a set of signals obtained from machining operations. Some of these signals are directly recorded from the machine (raw data) and filtered afterwards, while others are calculated from the raw data during preprocessing (clean data) (Tlusty, 2000); (Boothroyd; Knight, 1989).

2.2.2.1 Spindle Load

Spindle load is the instantaneous torque or power exerted on the spindle motor during machining measured in Newton meters (Nm). It reflects the resistance encountered by the cutting tool while removing material and can vary according to cutting depth, feedrate, tool geometry, and material properties.

2.2.2.2 Spindle Speed

Spindle speed represents the rotational velocity of the spindle, typically measured in revolutions per minute (RPM). This signal indicates how fast the cutting tool is rotating and is essential to understanding the kinematic conditions of the milling operation.

2.2.2.3 Feedrate

Feedrate is the linear speed at which the cutting tool advances relative to the workpiece along its toolpath. It is usually measured in millimeters per minute (mm/min) and directly affects the thickness of the material being removed per pass.

2.2.2.4 Chip Contact Area

Chip contact area is a calculated signal representing the actual surface area of the cutting tool that is in contact with the workpiece at any given time. It is derived from tool geometry, depth of cut, and engagement with the material, and varies along complex tool paths.

2.2.2.5 Chip Depth

Chip depth, is the calculated perpendicular thickness of the material layer removed by the cutting edge during a single pass. It depends on feedrate, spindle speed, and tool geometry, and it changes along curved or angled surfaces.

2.2.2.6 Forces

Forces are calculated vectors representing the magnitude and direction of the cutting forces acting on the tool during machining. They include components along the X, Y, and Z axes and are derived from load measurements, tool geometry, and chip formation characteristics.

2.2.2.7 Coordinates

Coordinates are the spatial positions of the cutting tool in three-dimensional space (X, Y, Z). They represent the actual trajectory of the tool relative to the workpiece and are essential for mapping calculated signals onto the machining path.

2.2.2.8 Engagement Time

Engagement time is a calculated signal that indicates the periods during which the cutting tool is actively in contact with the material. It is derived from spindle load, tool path, and feedrate, and reflects the active cutting versus idle motion along the toolpath.

2.3 STATISTICAL METHODS

This section describes the statistical methods applied within the framework to analyze machining signals and extract meaningful insights. These methods help identify relationships between variables, summarize data distributions, and ensure comparability across different scales (Smith, 1997).

2.3.1 Correlation

Correlation measures the strength and direction of the linear relationship between two variables. In this work, the **Pearson correlation coefficient** is used, defined as:

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{1}$$

where X_i and Y_i are individual observations of variables X and Y, and $\bar{X}$ and $\bar{Y}$ are their respective means. The correlation coefficient r_{XY} ranges from -1 to 1 (the closer to 1 and -1, the stronger it is), indicating negative (as one variable increases, the other tends to decrease), positive (as one variable increases, the other tends to increase), or no linear relationship.

2.3.2 Median

The median is a measure of central tendency that represents the middle value of a dataset when the observations are ordered. For a dataset $\{x_1, x_2, \ldots, x_n\}$, the median is given by:

$$\text{Median}(x_1, \ldots, x_n) = \begin{cases} x_{\frac{n+1}{2}}, & \text{if } n \text{ is odd} \\ \dfrac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}, & \text{if } n \text{ is even} \end{cases} \tag{2}$$

The median is particularly useful for machining signals because it is robust to outliers and non-symmetric distributions.

2.3.3 Normalization of Variables

Normalization is applied to scale different variables to a common range, making them comparable and improving the stability of subsequent analyses. In this work, min-max normalization is used:

$$X_i^{\text{norm}} = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \tag{3}$$

where X_i is the original value, X_{min} and X_{max} are the minimum and maximum values of the variable. The normalized variable X_i^{norm} ranges between 0 and 1 making it easier to visualize variables with very distant values.

2.4 FILTERS

In signal processing, filters are essential tools for removing noise, extracting relevant information, and improving the interpretability of data. Machining process signals often contain high-frequency noise or fluctuations caused by sensor dynamics, vibrations, or transient cutting events. Applying filters allows these signals to be smoothed or separated into frequency components, revealing the underlying trends and physical behavior of the process (Lyons, 2011).

Several types of filters can be applied depending on the purpose of the analysis:

2.4.1 Low-Pass Filters

Low-pass filters allow low-frequency components of a signal to pass through while attenuating higher frequencies. In machining data, they are particularly useful for isolating the slow variations in spindle load, feedrate, or forces that correspond to actual process trends, while suppressing high-frequency noise caused by vibration or measurement artifacts. A simple low-pass filter can be implemented through convolution with a smoothing kernel or by using a digital filter defined by a cutoff frequency f_c.

2.4.2 Moving Average Filters

The moving average filter is one of the simplest and most commonly used smoothing methods. It replaces each data point by the average of its neighboring values within a fixed window of size N:

$$y_i = \frac{1}{N} \sum_{k=0}^{N-1} x_{i-k} \tag{4}$$

where x_i is the original signal and y_i is the filtered signal. This approach reduces random fluctuations and highlights long-term trends in the data. Although effective for smoothing, it may slightly delay signal features and blur rapid transitions.

2.4.3 Gaussian Filter

The Gaussian filter is a smoothing technique based on the properties of the normal (Gaussian) distribution. It is used to reduce noise and smooth discrete data, such as histograms, by weighting nearby points according to a Gaussian function centered on each evaluation point. This approach preserves the general shape and trends of the data while attenuating local fluctuations caused by measurement variability or discretization.

Mathematically, the Gaussian filter is based on the Gaussian function:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{5}$$

where:

- μ is the mean (center) of the Gaussian distribution,
- σ is the standard deviation (which controls the width of the smoothing window),
- x is the variable being evaluated.

When applied to a dataset or histogram, the Gaussian filter can be interpreted as a convolution of the discrete data $f(x)$ with the Gaussian kernel $G(x)$:

$$(f * G)(x) = \int_{-\infty}^{\infty} f(t)\, G(x-t)\, dt \tag{6}$$

This operation replaces each data point with a weighted average of its neighbors, where the weights are determined by the Gaussian function. The result is a smooth curve that approximates the underlying probability density of the data.

In the context of histograms, this method is equivalent to a Gaussian kernel density estimation (KDE), where each data sample contributes a Gaussian distribution centered at its value. The resulting smoothed function $\hat{f}(x)$ is given by:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{7}$$

where:

- n is the number of samples,
- h is the bandwidth parameter controlling the degree of smoothing,
- $K(\cdot)$ is the Gaussian kernel.

The Gaussian filter, therefore, enables the generation of a smooth and continuous curve from discrete histogram data, allowing clearer visualization of underlying trends and distributions while minimizing the effects of random noise (Gonzalez; Woods, 2008).

2.4.4 Savitzky–Golay Filtering

The Savitzky–Golay (SG) filter is a digital smoothing technique based on local polynomial least-squares approximation. Unlike simple averaging filters, it preserves important signal features such as peak height, width, and curvature, making it particularly suitable for the analysis of measurement data where geometric fidelity is required.

Consider a discrete signal $y[n]$ sampled at uniform intervals. Within a sliding window of length $N = 2M + 1$ centered at sample n_0, the signal is locally approximated by a polynomial of degree p:

$$y[n_0 + k] \approx \sum_{j=0}^{p} a_j k^j, \quad k = -M, \dots, M \tag{8}$$

The polynomial coefficients $\mathbf{a} = [a_0, a_1, \dots, a_p]^T$ are obtained by minimizing the least-squares error:

$$\min_{\mathbf{a}} \sum_{k=-M}^{M} \left(y[n_0 + k] - \sum_{j=0}^{p} a_j k^j \right)^2 \tag{9}$$

This optimization problem can be written in matrix form as:

$$\hat{\mathbf{a}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \tag{10}$$

where $\mathbf{X}$ is a Vandermonde matrix constructed from the local coordinates k, and $\mathbf{y}$ contains the signal samples within the window.

The smoothed signal value at the central sample n_0 corresponds to the constant term of the fitted polynomial:

$$\hat{s}[n_0] = \hat{a}_0 \tag{11}$$

Due to the linearity of the least-squares solution, this operation can be expressed as a convolution:

$$\hat{s}[n_0] = \sum_{k=-M}^{M} h_k \, y[n_0 + k], \tag{12}$$

where the coefficients h_k depend solely on the window length N and polynomial degree p. Consequently, the SG filter acts as a fixed linear filter.

An important advantage of the Savitzky–Golay filter is its ability to estimate signal derivatives. The d-th derivative at the window center is obtained directly from the polynomial coefficients:

$$\hat{s}^{(d)}[n_0] = d! \, \hat{a}_d \tag{13}$$

This property enables simultaneous smoothing and differentiation, which is particularly useful in signal analysis applications.

In summary, the Savitzky–Golay filter performs local polynomial regression to reduce noise while preserving the underlying signal structure, providing superior shape preservation compared to conventional moving average filters (Savitzky; Golay, 1964).

2.5 FOURIER TRANSFORM

The Fourier Transform is a mathematical tool that decomposes a signal into its constituent frequencies. It provides insight into the frequency-domain representation of a time-domain signal, allowing for the analysis of periodic components and their amplitudes. For a continuous signal $x(t)$, the Fourier Transform is defined as:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\, dt \tag{14}$$

Here, $X(f)$ is the Fourier Transform of $x(t)$, f is the frequency, and j represents the imaginary unit. This equation transforms a time-domain signal into its frequency-domain counterpart, revealing the amplitude and phase of each frequency component.

The inverse Fourier Transform allows for the reconstruction of the time-domain signal from its frequency-domain representation:

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}\, df \tag{15}$$

For discrete-time signals, the Discrete Fourier Transform (DFT) is used, defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \tag{16}$$

where N is the number of samples, $x[n]$ is the discrete signal, and k indexes the frequency components (Oppenheim; Willsky Alan S. with Nawab, 1999).

2.5.1 Fast Fourier Transform - FFT

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT) and its inverse. The computational complexity of the direct DFT is $O(N^2)$, where N is the number of points in the signal. The FFT reduces this complexity to $O(N \log N)$, making it a practical choice for processing large datasets.

The FFT exploits the symmetry and periodicity properties of the complex exponential terms in the DFT. By recursively breaking down a DFT of size N into smaller DFTs of size $N/2$, the FFT achieves significant computational savings. This is particularly useful in applications such as signal processing, image analysis, and spectral estimation.

One of the most commonly used FFT algorithms is the Cooley-Tukey algorithm. It rearranges the input data into bit-reversed order and performs calculations in a divide-and-conquer manner. (NTi Audio, 2026)

2.5.2 Short-Time Fourier Transform - STFT

While the Fourier Transform provides a global frequency representation, it lacks temporal resolution. The Short-Time Fourier Transform (STFT) addresses this limitation by analyzing a signal in small time windows, providing a time-frequency representation. The STFT is defined as:

$$X(t, f) = \int_{-\infty}^{\infty} x(\tau) w(\tau - t) e^{-j2\pi f \tau} \, d\tau \tag{17}$$

Here, $w(\tau - t)$ is a window function centered at time t, which isolates a segment of the signal for analysis. Common window functions include the Hamming, Hanning, and Gaussian windows. The choice of window size involves a trade-off between time and frequency resolution.

The STFT produces a spectrogram, which visualizes how the signal's frequency content evolves over time. It is widely used in audio signal analysis, speech processing, and vibration analysis for its ability to capture transient and non-stationary characteristics of signals. (O'Gorman, 2026)

2.6 FREQUENCY COMPONENTS IN MACHINING

In machining processes such as milling, the frequency spectrum of measured signals (e.g., spindle load, cutting forces, or vibrations) provides valuable insights into the dynamic behavior of the operation. When applying Fourier-based methods such as the Fast Fourier Transform (FFT) or the Short-Time Fourier Transform (STFT), characteristic frequencies corresponding to machine elements, tool geometry, and process dynamics become visible. These frequency components can be associated with specific physical sources within the milling system, allowing for the identification of normal operation and the detection of chatter, imbalance, or tool wear (Rubio; Jáuregui-Correa, 2024).

The main frequency components typically observed in milling operations include:

- **Spindle Rotational Frequency (f_S)**: The fundamental frequency corresponding to the spindle rotation speed, given by

$$f_S = \frac{N_S}{60} \tag{18}$$

where N_s is the spindle speed in revolutions per minute (RPM). This frequency represents one full revolution of the spindle.

- **Tooth Passing Frequency (f_t)**: The dominant component in milling, corresponding to the frequency at which the tool teeth engage with the workpiece. It is calculated as

$$f_t = Z \cdot f_s \tag{19}$$

where Z is the number of cutting edges (teeth) on the tool. Harmonics of this frequency often appear in the spectrum due to periodic cutting forces.

- **Machine Structural Frequencies**: Natural frequencies associated with the spindle, tool holder, or machine structure. These appear as resonant peaks and are critical in chatter detection, as they can be excited by harmonics of the cutting process.

- **Chatter Frequency (f_c)**: A self-excited vibration frequency caused by the regenerative effect between the current and previous tool passes. It often appears near or between structural natural frequencies and may shift with cutting conditions.

- **Tool Imbalance or Runout Frequencies**: Components close to the spindle rotational frequency or its subharmonics, related to tool misalignment, imbalance, or eccentricity. These can indicate mechanical issues in the tool setup.

- **Feed-related Frequencies**: Lower-frequency components that correspond to feed motion irregularities or fluctuations in feedrate, typically much lower than the spindle-related frequencies.

- **External or Environmental Vibrations**: Frequencies not directly related to the machining process but introduced by external sources such as nearby equipment, motor drives, or floor vibrations.

Analyzing these frequency components allows for identifying the origin of dynamic phenomena in the milling process, enabling the differentiation between forced vibrations, self-excited chatter, and structural resonances.

2.7 PRIORITY ORDERING

To identify operations with the greatest potential for optimization, a priority score is computed from the total operation time and the engaged time. The method implemented in the software follows two steps: a raw priority metric that penalizes long operations with low engagement, and a min–max normalization to produce a bounded priority score in $[0, 1]$.

The raw priority metric is defined as

$$P_i^{\text{raw}} = \frac{T_i}{E_i + \varepsilon}, \tag{20}$$

where

- P_i^{raw} is the unnormalized priority for operation i,
- T_i is the total operation time for operation i (e.g. in minutes),
- E_i is the engaged time measure for operation i (expressed as a percentage or fraction of time spent cutting),
- ε is a small positive constant (e.g. 10^{-3}) introduced to avoid division by zero when E_i is zero or very small.

The raw metric (20) yields higher values for operations that have large total time but low engagement, highlighting inefficient operations. To make the priorities comparable and bounded, the raw scores are scaled using min–max normalization:

$$P_i = \frac{P_i^{\text{raw}} - \min_k P_k^{\text{raw}}}{\max_k P_k^{\text{raw}} - \min_k P_k^{\text{raw}}}, \tag{21}$$

with the usual guard that if $\max_k P_k^{\text{raw}} = \min_k P_k^{\text{raw}}$ then P_i is set to a default (e.g., 0 for all operations).

In the implemented pipeline the steps are:

1. Read and cast T_i and E_i as floats.
2. Compute P_i^{raw} using equation (20) with $\varepsilon = 10^{-3}$.
3. Apply min–max scaling to obtain $P_i \in [0, 1]$ (this is equivalent to using a 'MinMaxScaler').
4. Sort operations by P_i in descending order to obtain the final prioritized list.

This approach is robust and simple to interpret: operations with large total time and small engaged percentage receive the highest priority for investigation and optimization (Kendall, 1962).

2.8 TOOLS USED FOR DEVELOPMENT

2.8.1 Application Programming Interface (API)

An Application Programming Interface (API) defines a set of rules and communication protocols that allow different software systems to exchange data and interact with one another. In essence, an API acts as an intermediary that enables one application to access specific functionalities or datasets from another without needing direct access to its internal code or structure. This abstraction facilitates modularity, reusability, and scalability across complex systems.

2.8.1.1 Representational State Transfer (REST)

Representational State Transfer (REST) is an architectural style commonly used for designing networked APIs based on standard HTTP methods such as `GET`, `POST`, `PUT`, and `DELETE`. RESTful APIs communicate using stateless requests, where each message from the client to the server contains all the information needed to process the request. The exchanged data is typically structured in lightweight formats such as JSON, making it both human-readable and efficient for machine parsing (Richardson; Amundsen; Ruby, 2020).

In this project, the software infrastructure already includes two REST APIs that serve as the main data interfaces for the analysis framework. The first API provides access to overall process and test data, including measurements, time-series signals, and metadata. The second API manages the information related to tool assemblies, containing details about tool geometry, materials, and configuration parameters. These APIs enable seamless integration between the analytical modules, the frontend interface, and the existing digital twin ecosystem, ensuring efficient and consistent data flow throughout the system.

2.8.2 Docker

Docker is an open-source platform designed to automate the deployment and management of applications within lightweight, portable containers. Each container includes the necessary code, runtime, system tools, and libraries required for execution, ensuring that applications run consistently across different environments. Unlike traditional virtual machines, containers share the same operating system kernel, which makes them much more efficient in terms of performance and resource utilization (Bozman, 2024).

In this project, Docker was used to create a modular and reproducible development environment. Its practical modularity allows the system to be divided into independent components—such as the data processing backend, database, and user interface—each running in its own container. This separation simplifies maintenance, testing, and scalability, as each service can be updated or replaced without affecting the others. Additionally, Docker ensures environment consistency between development, testing, and production stages, reducing integration issues and improving the reliability of the overall framework.

2.8.3 Python

Python is a versatile, high-level programming language widely used in data analysis and scientific computing. Its simple syntax and readability make it accessible to beginners while remaining powerful for advanced users. The language offers an extensive ecosystem of libraries, such as `pandas`, `numpy`, and `plotly`, which provide robust tools for data manipulation, numerical computations, and visualization. Python seamlessly integrates with various file formats, including JSON, CSV, Excel, and databases, enabling efficient data handling. It is platform-independent, allowing for consistent execution across different operating systems, and can scale from small-scale projects to handling large datasets. With a vast and active community, Python benefits from continuous development, extensive resources, and a wide range of tools, making it an indispensable cornerstone of modern data analysis and visualization (Lutz, 2013).

2.8.3.1 Streamlit

Streamlit is an open-source Python framework designed for creating interactive web applications for data visualization and analysis. It enables developers to transform Python scripts directly into shareable web interfaces with minimal effort, without requiring traditional frontend development skills such as HTML, CSS, or JavaScript. Streamlit's declarative syntax and component-based structure allow for rapid prototyping and intuitive user interaction with analytical results (Streamlit Community, 2026).

In this project, Streamlit was used to develop the frontend interface of the framework. Its interactive elements—such as sliders, buttons, file uploads, and plot containers—provide an accessible and user-friendly way for engineers to explore machining data, visualize process parameters, and navigate through different analysis modules. By integrating seamlessly with the backend APIs and analytical algorithms, Streamlit ensures that complex technical insights are presented in a clear and intuitive manner.

2.8.3.2 Plotly

Plotly is a powerful Python library for creating interactive and high-quality data visualizations. It supports a wide range of chart types, from basic line and scatter plots to complex 3D surfaces and multi-axis time series. Plotly's interactivity enables zooming, hovering, and filtering of graphical elements, allowing users to explore large datasets and dynamic relationships visually (Fortunato, 2023).

Within this framework, Plotly was employed to generate graphical representations of machining process data, including time series plots of spindle speed, load, feedrate, and frequency spectra derived from FFT and STFT analyses. It also supports 3D path visualizations that display tool trajectories, critical load zones, and engagement points. These interactive visualizations enhance the interpretability of the results and enable a more effective understanding of the process behavior over time.

2.8.3.3 Libraries

2.8.4 MongoDB

The gemineers software uses MongoDB as its document-oriented database, opting for a NoSQL and non-relational scheme for the application. Unlike traditional relational databases with their rigid structures, MongoDB stores data in JSON-like documents, making it perfect for complex and ever-changing information (MongoDB, 2026). As mentioned in the subsections 3.1.1 and 3.1.2 some of the data and after-simulation information are stored in JSON files, so having a database that can store this in similar documents makes the process easier and faster. MongoDB also offers schema flexibility, so you can define structures for consistency while allowing some variation within your data. And as your data collection grows, MongoDB can efficiently handle it by distributing information across multiple servers. This, combined with its speed for reading and writing data, makes MongoDB a perfect choice for the gemineers software, that demands high performance.

2.8.5 JavaScript Object Notation (JSON)

The JSON (JavaScript Object Notation) is an open standard, programming language independent, human readable, lightweight data interchange file format and easy for machines to parse and generate (Gouy, 2006). All data gathered from the DAQ system is saved in this format in the Mongo database, which means that all data analysis were made from this data format. In Figure 5 is an example of one JSON file received from the DAQ.

Figure 5 – JSON format example


```json
{
    "timestamp": {
        "unit": "ms",
        "reference_format": "unix_timestamp",
        "values": [
            1727184000122
        ]
    },
    "ToolName": [
        "MSK62_7"
    ],
    "DuploNumber": [
        1
    ]
}
```


Source: Company‘s archive

3 DESCRIPTION OF THE PROBLEM AND TECHNICAL REQUIREMENTS

3.1 SOFTWARE CAPABILITIES

The gemineers software platform is built upon four primary pillars that will be discussed bellow, also an image explaining the platform can be observed in the image **??**:

3.1.1 Data Ground or Data Acquisition

The Data Acquisition segment is responsible for gathering data directly from the machine tools during machining operations, marking the initial phase in the overall process chain that subsequent software areas will adhere to. Its duty encompasses converting acquired data into comprehensible information and standardizing various data types.

3.1.2 Data Processing

The data processing area is responsible for converting the acquired data into useful information. This involves running simulations, performing mathematical analyses, and extracting insights. This area comprises several services, each handling a specific aspect of data transformation and predictive modeling.

3.1.3 Back end or REST API

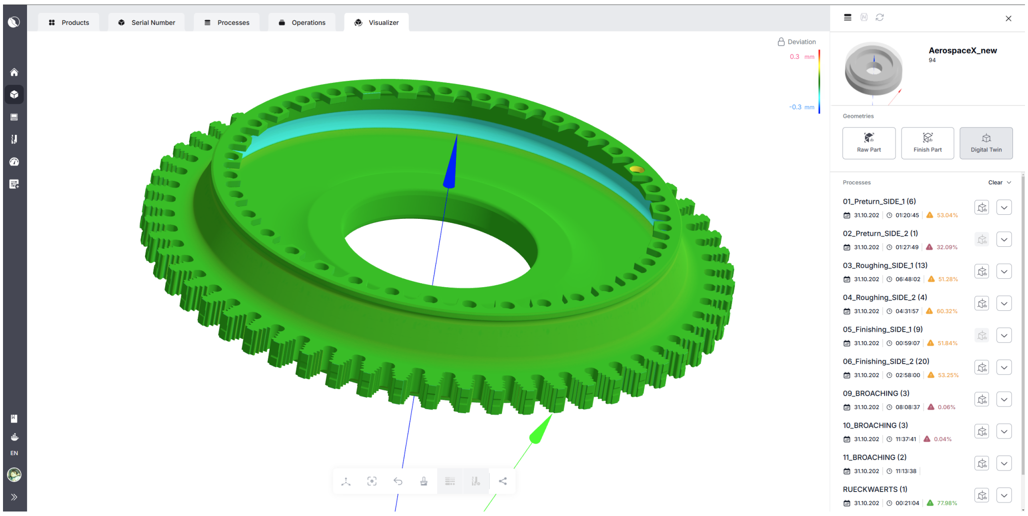
This service is responsible for harmonizing other services and domains, taking charge of database management and user authentication/authorization within the software.

3.1.4 Front end

The front-end of the company's software serves as a critical user interface, providing operators and engineers with real-time access to essential machining data. It displays live data collected directly from CNC machines, offering insights into the current state of the manufacturing process. Additionally, the front-end presents processed information, including tool wear metrics and the digital twin of the manufactured part, enabling users to monitor and analyze production performance with precision. This intuitive interface bridges the gap between raw machine data and actionable insights, enhancing decision-making and process optimization.

The digital twin displayed on the software provides a color-coded representation of the product's surface: areas in green indicate that the product meets quality specifications, blue signifies material removal beyond the desired limit (indicating excessive cutting), and red highlights areas here excess material remains (indicating insufficient cutting), which can be seen on Figure 6.

Figure 6 – Digital twin page



Source: Company‘s archive

3.2 COST TIME AND QUALITY PARAMETERS

In modern manufacturing environments, particularly within the aerospace and automotive sectors, the optimization of cost, time, and quality parameters represents a fundamental objective. These three dimensions form the basis of what is commonly referred to as the manufacturing performance triangle or the project management “iron triangle.” In this relationship, improvements in one dimension often influence the others, emphasizing the need for a balanced and data-driven approach to process optimization.

The **cost** parameter is primarily associated with resource consumption during production, including tool wear, energy usage, and machine occupancy time. Reducing unnecessary machining operations, tool breakage, or inefficient parameter combinations directly translates into cost savings.

The **time** parameter is linked to overall productivity and cycle efficiency. It includes the active machining time, tool engagement duration, and idle times such as air cuts or tool changes. Identifying and minimizing non-productive periods allows for a higher throughput and better utilization of equipment.

The **quality** parameter reflects the integrity and precision of the manufactured part, strongly influenced by cutting forces, vibrations, tool condition, and machine stability. Quality-related analyses often rely on force and frequency measurements to detect undesirable conditions such as chatter, excessive load, or thermal deformation, which can degrade surface finish or dimensional accuracy.

By quantitatively evaluating these three parameters, it becomes possible to identify correlations and trade-offs that guide decision-making for process improvement through algorithmic and statistical tools that deliver actionable insights into cost efficiency, time optimization, and quality assurance.

3.3 HISTORICAL BACKGROUND ON QUALITY INSPECTION

The evolution of manufacturing quality management has been closely tied to the advancement of measurement, inspection, and statistical control techniques. Early quality control practices in the industrial era were largely based on post-process inspections, where parts were manually measured and compared against predefined tolerances. While this approach ensured conformance, it was time-consuming, reactive, and inefficient for high-volume production especially in industries that demand higher standards such as automotive and aerospace.

Incidents involving aero engine components have highlighted the importance of robust manufacturing practices and thorough quality control. For instance, investigations into specific aero engine failures have revealed manufacturing-induced anomalies as contributing factors. The National Transportation Safety Board (NTSB) reports provide clear examples:

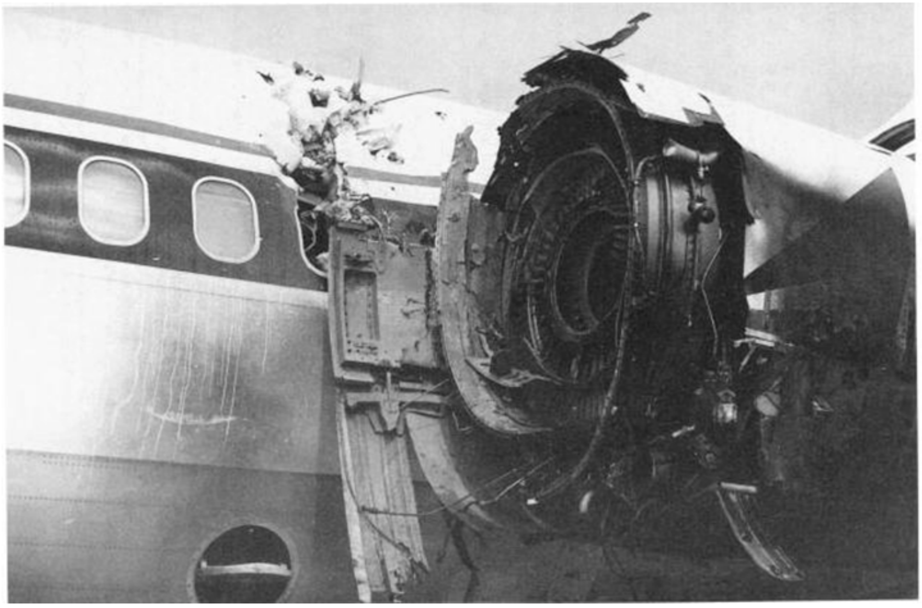
- DCA96MA068 (Pensacola, July 6, 1996): "Some form of drill breakage or drill breakdown, [. . .], occurred during the drilling process, creating the altered microstructure and ladder cracking in the accident fan hub." (NTSB 1998)
- ENG14IA028 (Long Beach, September 18, 2014): "During a machining operation of the disc lug, a tool mark was introduced that set up the area for fatigue cracks to initiate." (NTSB 2016)
- DCA21FA085 (Broomfield, February 20, 2021): "Examination [. . .] revealed a discontinuity in a local tight radius in the internal blade geometry that had been introduced during the machining [. . .] reducing the fatigue life by 50

These examples emphasize the critical need for rigorous quality assurance measures and adherence to manufacturing best practices to prevent defects that can compromise structural integrity and operational safety.

3.4 TECHNICAL REQUIREMENTS

The technical requirements define both the functional and non-functional aspects necessary for the successful development and deployment of the proposed framework. These requirements are organized into two levels: overall requirements, which apply to the entire system, and specific requirements for each test module.

Figure 7 – Uncontained failure due to a manufacturing induced anomaly



Source: (Wikipedia contributors, 2026)

3.4.1 Overall Requirements

3.4.1.1 Functional Requirements

- Ensure full compatibility and seamless integration with the existing Digital Twin infrastructure.
- Enable automatic data extraction, cleaning, and preprocessing immediately after a test execution.
- Integrate statistical, correlation, and signal processing algorithms into the backend for automated analysis.
- Provide visualization of results through an intuitive, interactive, and unified web-based interface.

3.4.1.2 Non-Functional Requirements

- Ensure scalability to accommodate different test types, machines, and extensions.
- Guarantee maintainability through modular code structure and containerization (Docker-based).
- Provide performance efficiency with responsive data loading and visualization times under large datasets.

- Ensure usability through a clear, intuitive, and responsive interface.
- Guarantee data integrity and reliability through validation checks during data import and preprocessing.

3.4.2 Cost and Time Analysis Module

3.4.2.1 Functional Requirements

- Allow selection of the tool (Gemini) and serial number for analysis.
- Prioritize operations based on total time and time engaged using the priority ordering algorithm.
- Allow selection of a specific range of operations for focused analysis.
- Support analysis for both end mill and ball mill tools.
- Calculate flute length usage for end mills and radial engagement angle for ball mills.
- Detect and classify feedrate modes and correlate them with flute length utilization to assess intentional underperformance.
- Generate statistical visualizations displaying engaged vs. non-engaged time, cutting contact, and correlation metrics.
- Display normalized feedrate vs. contact graphs and highlight warning zones.
- Provide actionable recommendations to improve machining efficiency based on feedrate and engagement behavior.
- Estimate potential time savings if optimal feedrate settings were applied.

3.4.2.2 Non-Functional Requirements

- Ensure visual clarity and interactivity in feedrate and time plots.
- Maintain computational efficiency for large operation datasets.
- Guarantee consistent formatting and exportability of generated reports.

3.4.3 Quality Analysis Module

3.4.3.1 Functional Requirements

- Allow selection of the tool (Gemini) and serial number for analysis.
- Enable selection of specific operation ranges for detailed investigation.

- Display load and force data across the toolpath.
- Automatically detect and highlight high-load peaks that may indicate tool wear or breakage.
- Display a 3D toolpath visualization color-mapped according to load magnitude and warning regions.

3.4.3.2 Non-Functional Requirements

- Ensure smooth 3D rendering performance with real-time interactivity.
- Maintain high color contrast and clarity in visualizations.
- Allow export of critical peak data for external inspection.

3.4.4 Frequency Analysis Module

3.4.4.1 Functional Requirements

- Allow selection of data either from the database or through external file upload.
- Enable users to select operations and specific signals for frequency analysis.
- Allow filtering by frequency range and magnitude thresholds.
- Display 2D plots for FFT, STFT, and filtered signals.
- Provide a 3D toolpath visualization where each point is color-mapped by the corresponding signal amplitude (filtered or unfiltered).

3.4.4.2 Non-Functional Requirements

- Ensure fast computation of FFT/STFT transformations with optimized processing.
- Allow interactive hovering and zooming across all graphs.
- Guarantee consistent color mapping and scaling across frequency plots.

3.4.5 Time Series Visualization Module

3.4.5.1 Functional Requirements

- Display the list of available operations stored in the database.
- Allow selection of one or more operations for simultaneous visualization.
- Allow selection of one or multiple signals to be plotted concurrently.

- Provide the option to visualize raw data, clean data, or both for comparison.
- Enable display modes for overlapping or stacked time series plots.

3.4.5.2 Non-Functional Requirements

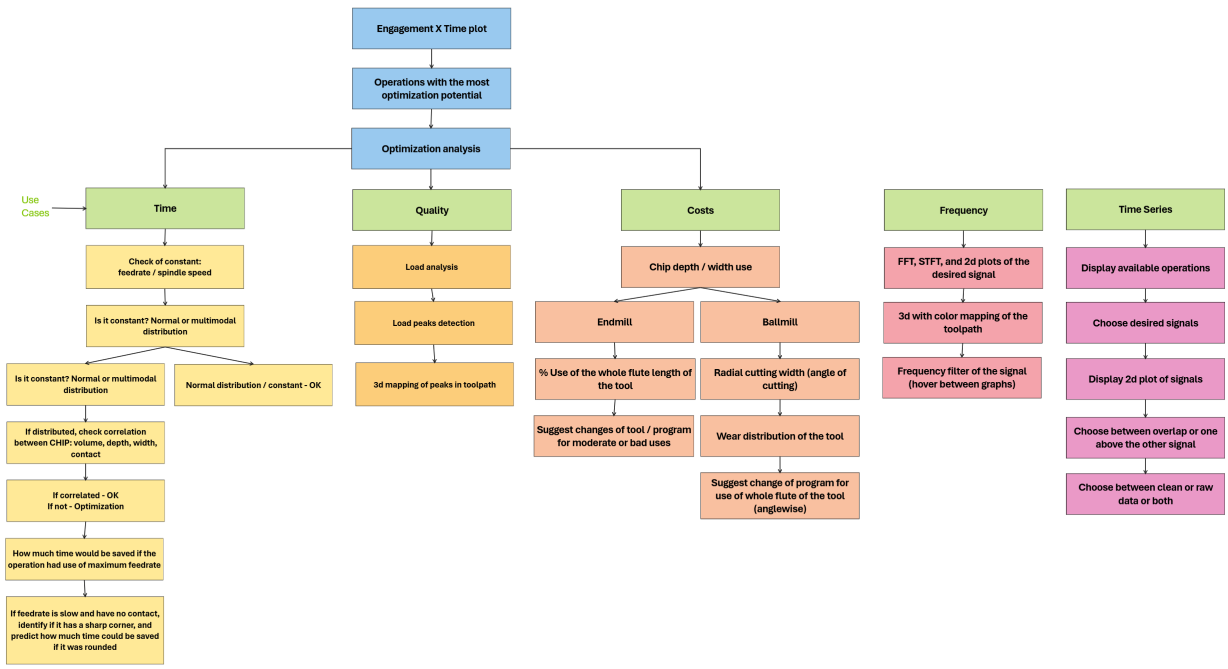
- Maintain visual responsiveness even when plotting multiple large datasets.
- Ensure synchronized axes and legends across subplots.
- Provide high export quality for plots in image or HTML format.

4 PROPOSED SOLUTION AND METHODOLOGY

4.1 SYSTEM ARCHITECTURE

The system is a modular Python framework for machining process analysis, organized around five analysis domains (use cases): cost, time, quality, frequency and time series, as shown in the Figure 8.

Figure 8 – Application Fluxogram



Source: Author‘s archive

Each module operates independently but follows a unified workflow that begins with the identification of machining operations with the highest optimization potential. This identification is based on the relationship between total machining time and tool engagement time, which acts as a preliminary filter to determine where analytical efforts should be concentrated. From this point, the framework branches into specialized analysis paths, depending on the selected domain.

- **Time Analysis Module**: Evaluates spindle speed and feedrate behavior to determine whether machining parameters remain constant or exhibit variations. When deviations are found, the module examines correlations with chip parameters such as volume, depth, width, or tool–workpiece contact. It also estimates potential time savings by suggesting optimal feedrate usage and geometric simplifications (e.g., rounding sharp corners).

- **Quality Analysis Module**: Focuses on load behavior and force signals to detect high-load peaks, which may indicate tool wear, poor cutting conditions, or risk of tool breakage. These peaks are subsequently projected onto a 3D toolpath, enabling spatial localization of critical events within the machining trajectory.
- **Cost Analysis Module**: Divided into endmill and ball mill evaluations, this module analyzes tool usage efficiency by assessing flute length utilization, chip width/depth, and radial engagement. Inadequate usage is highlighted, and corrective actions such as toolpath changes or tool replacement are recommended to reduce cost per part.
- **Frequency Analysis Module**: Applies FFT, STFT, and filtering techniques to assess dynamic behavior of machining operations. The system allows visualization of dominant frequencies and maps these onto the 3D toolpath to identify regions affected by chatter or vibration-related issues.
- **Time Series Visualization Module**: Enables the user to select multiple operations and signals, display raw or processed data, and visualize them either overlapped or separated. This module serves as a general-purpose diagnostic tool for temporal signal inspection.

These analysis flows are implemented within an extensible architecture where each module shares the same data access layer (REST API), visualization engine (Streamlit + Plotly/3D path rendering), and preprocessing pipeline (normalization, filtering, interpolation). This modularity ensures that new tests or analytical models can be integrated without restructuring the existing system.

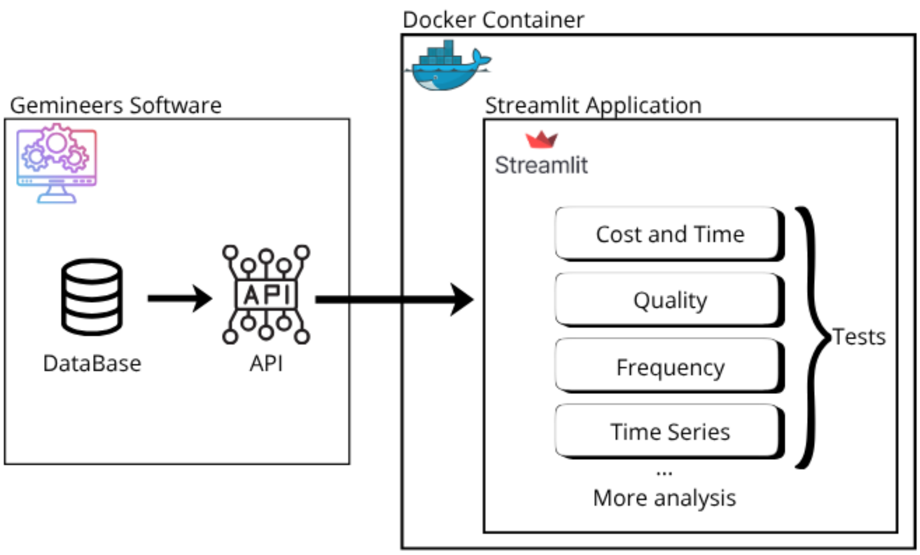
Following the modular organization of the analytical framework, Figure 9 illustrates how the system integrates with the existing Gemineers digital-twin infrastructure. The architecture is structured as a data-driven pipeline in which each component plays a distinct role within the analysis workflow.

At the base of the system is the Gemineers software environment, which contains the internal database where both raw and processed machining data are stored. This includes spindle signals, feedrate, forces, tool coordinates, chip geometry, and all other parameters required for the different analysis modules. Access to this information is facilitated by the company's existing REST API, which acts as an intermediary layer responsible for securely exposing selected data endpoints to external applications. Each of the modules uses different information provided by the API, therefore, each function on the backend returns a specific set of data in a dictionary format, which subsequently is joined by other necessary dictionaries in standardized variables format to fullfill the desired analysis.

The analytical platform developed in this project connects directly to this API. The entire analysis interface is deployed inside a Docker container, ensuring an isolated and reproducible environment. Within this container runs the Streamlit application, which houses all the use cases described previously—Cost and Time, Quality, Frequency, and Time Series analysis. Streamlit is responsible for managing user interaction, fetching data through the API, processing it using the backend algorithms, and presenting the results through dynamic plots, tables, 3D toolpaths, and interactive visualizations.

This architectural structure allows the analytical framework to remain fully decoupled from the main software platform while still operating seamlessly on top of its data. Docker ensures portability and simplifies deployment, the REST API guarantees consistent and controlled data access, and the Streamlit layer provides a unified, intuitive interface through which all analyses are executed. As a result, the complete system functions as an extension of the digital twin, adding analytical capabilities without interfering with existing infrastructure.

Figure 9 – Application Architecture



Source: Author‘s archive

4.2 SCRUM-BASED METHODOLOGICAL WORKFLOW

The development of the analytical framework followed a Scrum-inspired agile methodology, combining iterative progress, daily communication, weekly deliverables, and long-term Gantt planning (Drumond, 2026). This hybrid structure ensured both adaptability during development and adherence to strategic deadlines.

4.2.1 Scrum Practices Adopted

- **Daily Updates (Daily Scrum)** Short daily meetings were used to report completed tasks, plan next steps, and identify possible blockers, ensuring continuous alignment and rapid issue resolution.
- **Weekly Deliveries (Sprint Reviews)** Work was organized into weekly increments, each producing a functional or analytical deliverable. These increments were evaluated weekly, integrated into the framework, and refined as necessary.
- **Gantt Planning Integration** A long-term Gantt chart provided macro-level visibility of milestones and dependencies, while Scrum enabled flexibility and iterative development within each scheduled block.

4.2.2 Planning and Design Phase (4 Weeks)

This phase established the conceptual and analytical foundations of the framework. Each week corresponded to an independent sprint with clearly defined deliverables.

1. Week 1: Requirements Gathering
2. Week 2: KPIs and Analysis Definition
3. Week 3: Structure Definition
4. Week 4: Study of Algorithms

4.2.3 Execution Phase (9 Weeks)

This phase corresponded to the iterative construction of the system. Each analytical component was developed, validated, and integrated through weekly sprints.

1. Week 1: General Structure of the Framework
2. Weeks 2–3: Cost Analysis Module
3. Week 4: Time Analysis Module
4. Week 5: Quality Analysis Module
5. Weeks 6–7: Frequency Analysis Module
6. Week 8: Additional Time Series Analysis

4.2.4 Evaluation and Conclusion Phase

Several tasks in this phase ran in parallel. Weekly iterations were maintained, even for longer deliverables, to ensure gradual refinement.

1. Tests with Different Products (3 Weeks)
2. KPIs Analysis (3 Weeks)
3. Survey of Optimizations and Improvements (3 Weeks)
4. Report Writing (2 Months)
5. Presentation (2 Weeks)

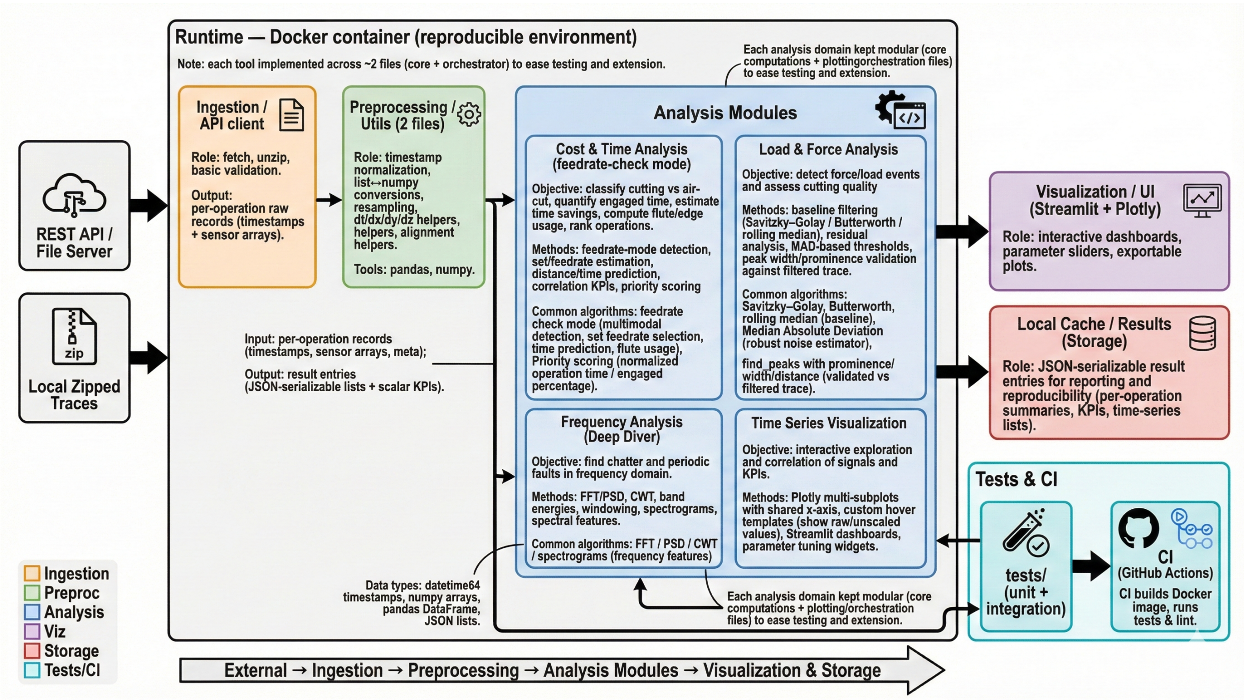
The next chapter describes how this application architecture and its related solutions are implemented. The gathered results are also presented and discussed within the following chapter.

5 DEVELOPMENT AND RESULTS

5.1 DETAILED IMPLEMENTATION OVERVIEW

This chapter presents the architectural design and implementation details of the project. The system follows a linear data flow pipeline—from ingestion to analysis and finally to visualization—encapsulated within a reproducible Docker runtime environment.

Figure 10 – Detailed Structure Overview



Source: Author‘s archive

The core components of the architecture illustrated in Figure 10 include:

- **Ingestion Layer (`utils/api_utils.py`):**
  This module acts as the entry point for raw data. It interfaces with external sources (REST APIs or local file servers) to fetch zipped trace files. It handles decompression, basic validation, and the parsing of raw JSON or binary structures into standardized per-operation records.

- **Preprocessing & Utilities (`utils/`):**
  Before analysis, raw data undergoes normalization. Key operations include timestamp synchronization, type conversion (optimizing Python `list` structures into `numpy` arrays for vectorization), and signal resampling. Helper functions for calculating derivatives (velocity, acceleration) and spatial metrics are also centralized here to ensure consistency across all analysis modules.

- **Analysis Modules (`tools/`):**
  The analytical core is divided into four distinct, isolated domains to maintain modularity:

  1. **Cost & Time Analysis:** Focuses on operational efficiency. It implements algorithms for feedrate mode detection (using Kernel Density Estimation and peak detection) to quantify air-cuts versus cutting time, estimate potential savings, and calculate flute usage KPIs.
  2. **Load & Force Analysis:** Assesses process stability. It utilizes robust filtering techniques (Savitzky–Golay, rolling median) and statistical thresholds (Median Absolute Deviation) to detect anomalies in force and load signals.
  3. **Frequency Analysis (Deep Diver):** A specialized module for detecting periodic faults such as chatter. It leverages Fast Fourier Transforms (FFT) and Short Time Fourier Transforms (STFT) to extract spectral features.
  4. **Time Series Visualization:** Handles the aggregation of signals for rendering, ensuring shared x-axes and synchronized hover templates for identifying correlations across different signal types.

- **Visualization & Storage:**
  The user interface is built using **Streamlit**, providing interactive dashboards that allow users to tune parameters and explore data visually via **Plotly**. Results and computed KPIs are serialized into JSON format and stored in a local cache, ensuring that computationally expensive analysis does not need to be re-run for reporting.

5.1.1 Main Application Router and Entry Point

The `main.py` file serves as the application's central entry point and router, orchestrating session state initialization, dynamic tool discovery, page registration and multi-page navigation.

Upon startup, the application initializes an authentication flag in session state:

$$\text{if } \texttt{"logged\_in"} \notin \text{st.session\_state} \Rightarrow \text{st.session\_state[“logged\_in”]} = \text{False.} \tag{22}$$

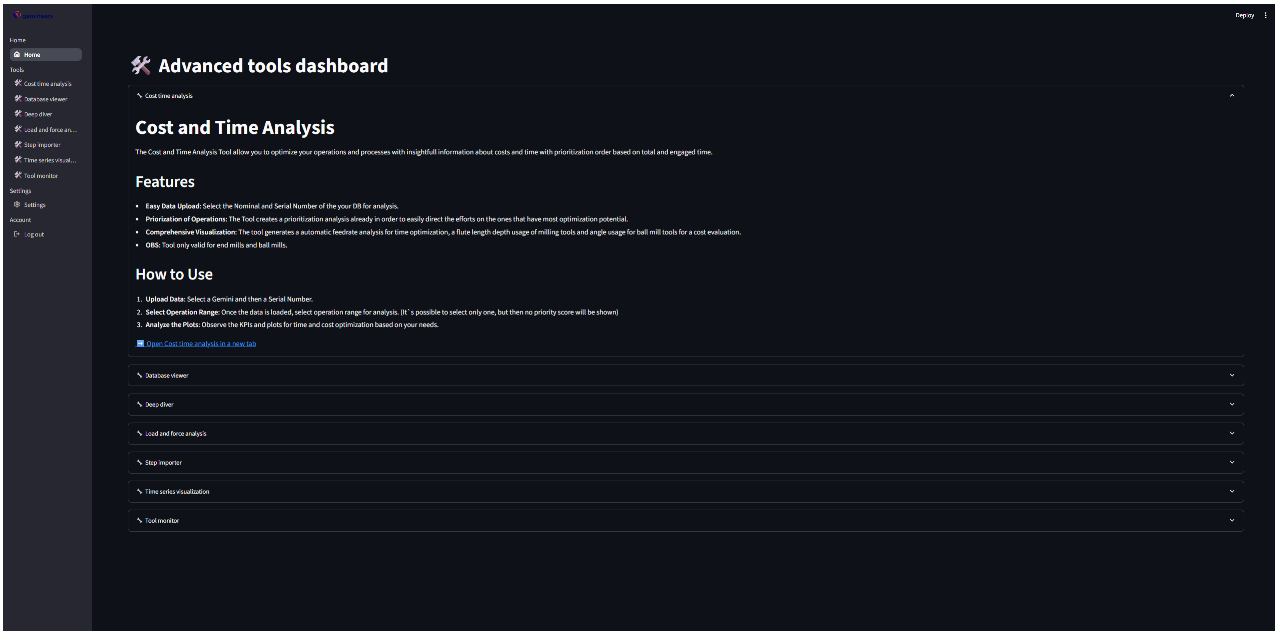
Two functions, `login()` and `logout()`, toggle this flag and trigger re-renders via `st.rerun()`. Currently, the authentication check is bypassed (conditional is `or True`), allowing unrestricted access to all pages.

Tool discovery occurs dynamically via `utils.get_available_tools()`, which scans the `tools/` directory and returns folder names and paths. For each folder, the application checks whether a matching entry-point Python file exists (e.g., `cost_time_analysis.py` in `tools/cost_time_analysis/`). Matching tools are registered as Streamlit pages and organized under the “Tools” section of the navigation menu. Once considered tools, a description of it is shown on the home page by reading a readme file from their folders.

The navigation structure is rendered as an interactive sidebar menu. Page transitions preserve session state, ensuring that data cached by the home page remains accessible to all downstream tools.

The dynamic tool discovery pattern enables extensibility: new analysis tools can be added by creating a new folder in `tools/` with a matching entry-point file, without modifying the router. The application is executed via `pg.run()`, which starts the multi-page Streamlit server shown in Figure 11.

Figure 11 – Home Page



Source: Author‘s archive

5.2 IMPLEMENTATION: SYSTEM ARCHITECTURE AND FRAMEWORK INTEGRATION

5.2.1 Architectural Overview

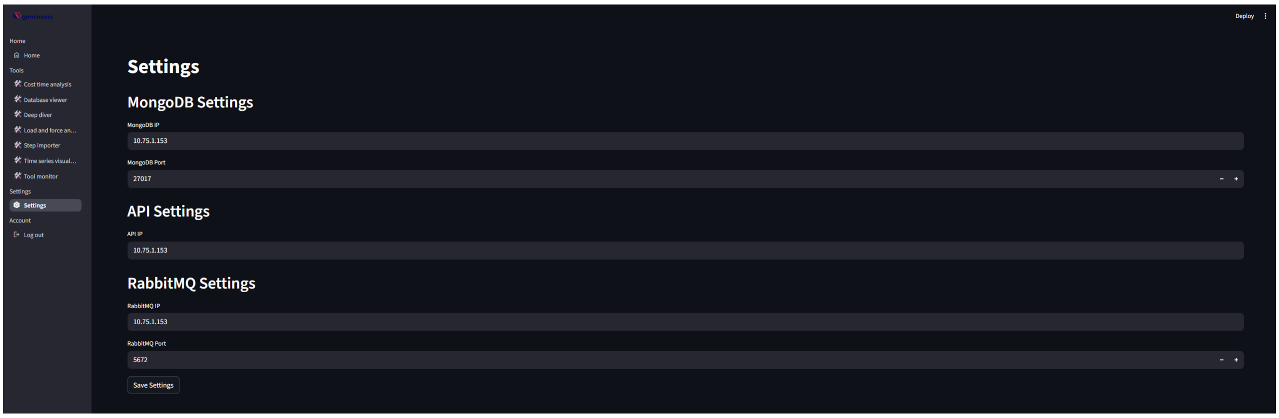
The framework is implemented as a modular, stateful web application designed to support interactive analysis of machining data. Each analysis capability—load and force analysis, frequency analysis, time-series visualization, and cost and time optimization—is implemented as an independent tool while sharing a common infrastructure for configuration, authentication, data access, and session management.

The application follows a layered architecture. A Streamlit-based presentation layer provides interactive dashboards, persistent user selections, and reactive visualizations. Beneath this, a tool layer encapsulates domain-specific logic and plotting routines for each analysis module. A shared data access layer abstracts communication with remote services, while utility components handle common tasks such as data parsing, normalization, and user-interface convenience functions. This design promotes separation of concerns, extensibility, and reuse across tools.

5.2.2 Configuration, Session Management, and Authentication

At startup, the application loads configuration parameters—such as API connection details and optional database or messaging service endpoints—from a centralized settings file. These parameters are stored in session state and remain available throughout the user session, avoiding repeated file access and ensuring consistent behavior across tools. The settings interface used to manage these parameters is shown in Figure 12.

Figure 12 – Settings Page



Source: Author‘s archive

Authentication and data access are handled through an API connector abstraction that manages login, session cookies, and request headers. Once authenticated, the connector transparently maintains the session for all subsequent data requests. Network or authentication failures are handled gracefully, with informative warnings presented to the user instead of application crashes, allowing continued interaction with already available data.

5.2.3 Data Retrieval and Normalization

The framework retrieves machining data through a structured, multi-step pipeline. Users first select a nominal and serial number, after which available processes and operations are indexed and presented for selection. For the chosen operations, the system downloads either raw data, cleaned data, or both. Operation data are delivered by the API as nested ZIP archives, which are automatically extracted and parsed into structured in-memory representations.

Once downloaded, the data are normalized into time-indexed formats suitable for analysis. Signal channels are aligned in time, optionally resampled to a constant frequency, and standardized in naming to ensure consistency across tools. This preprocessing stage ensures that all downstream analyses—regardless of tool or signal type—operate on coherent and comparable datasets.

5.2.4 Multi-Tool Navigation and Stateful Workflows

Tool navigation is dynamically generated by discovering available analysis modules at runtime, allowing new tools to be added without modifying the core application logic. Session state persistence enables seamless workflows in which data downloaded in one tool can be reused in another, reducing redundant API calls and improving responsiveness.

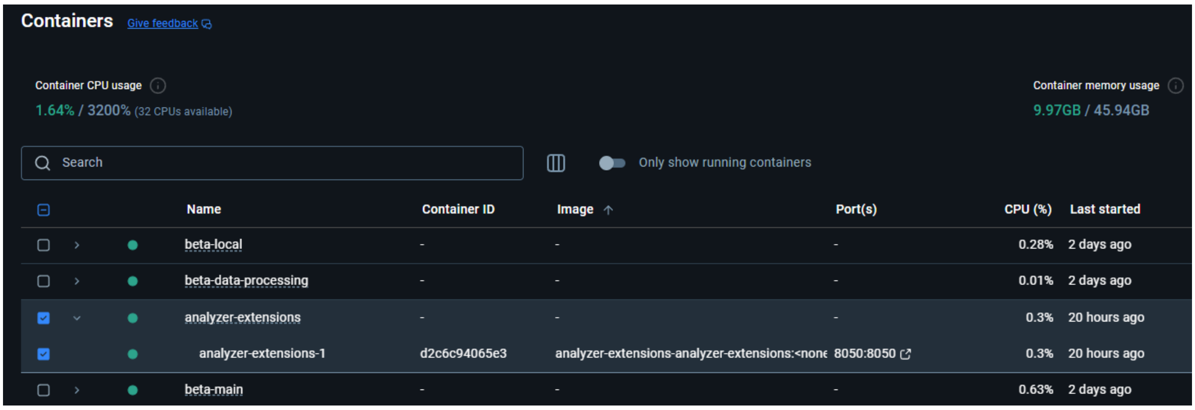
A typical workflow involves selecting operations, downloading data once, and then iteratively exploring the data across different analysis tools. Parameter adjustments within each tool trigger reactive updates, allowing rapid exploratory analysis while preserving previously computed results and user selections.

5.2.5 Performance, Robustness, and Deployment

To ensure responsiveness, the framework caches downloaded data and preprocessed results at the session level, minimizing repeated computation and network overhead. Defensive error handling and data validation checks prevent incomplete or malformed datasets from propagating into analysis routines, improving overall robustness.

The entire application is containerized using Docker, providing a reproducible and isolated runtime environment. Deployment is managed via Docker Compose, which defines port mappings, volume mounts for live code updates, and restart policies. This setup simplifies development, deployment, and integration with existing software ecosystems. An overview of the running analyzer container alongside other system containers is shown in Figure 13.

Figure 13 – Analyzer Containers Along With Software Containers Running



Source: Author's archive

5.3 IMPLEMENTATION: COST AND TIME ANALYSIS

5.3.1 Objective and Scope

The cost and time analysis module transforms per-operation telemetry and metadata into actionable indicators for process optimization. Its primary objective is to identify inefficiencies in tool usage and motion execution and to estimate the potential gains achievable through improved parameter selection and trajectory strategies.

The main outputs of the module include:

- separation of productive (cutting) and non-productive (air or inefficient) motion;
- quantification of tool utilization, expressed either as flute length usage (for end mills) or angular usage (for ball mills);
- identification of dominant feedrate regimes and inference of whether they are intentional or incidental;
- deterministic estimation of potential time savings under alternative feedrate or motion strategies;
- prioritization of operations according to their optimization potential.

5.3.2 Processing Pipeline

The module follows a linear and deterministic processing pipeline, designed to progressively refine raw operation data into interpretable key performance indicators (KPIs) and optimization suggestions.

Initially, operation bundles are validated to ensure that minimal metadata and signal availability requirements are met. Operations lacking sufficient information are excluded early, preventing unreliable downstream analysis. For valid operations, sample counts and signal availability are recorded to assess whether further statistical analysis is meaningful.

Next, all operations are globally ranked according to a priority score that combines absolute operation duration with the proportion of engaged cutting time. This ranking emphasizes long operations with low engagement, as these typically offer the highest absolute potential for time recovery.

For each prioritized operation, raw telemetry is converted into numerically consistent time series. Sampling characteristics are estimated and used to parameterize subsequent detectors and filters. Spatial trajectories are then processed to compute total traveled distance and cutting distance, while discrete counts—such as contact duration, low-utilization samples, and categorical feed/contact states—are accumulated to form utilization percentages and rule-based indicators.

Tool-specific usage modeling is applied depending on the cutter geometry. For end mills, axial and flute length usage metrics are computed. For ball mills, removed material is mapped onto a discretized cutter profile to estimate angular usage and derive a weighted average cutting angle, providing insight into how effectively the tool geometry is exploited.

Feedrate behavior is analyzed by isolating cutting samples and estimating the distribution of operating speeds. Dominant feed regimes are extracted and combined with correlation measures between feedrate and chip-related variables to infer whether multiple regimes result from deliberate operator settings or from process variability. Robust fallback logic ensures that degenerate cases (e.g., constant feed or sparse data) still yield conservative outputs.

Based on these results, the module deterministically estimates cutting time under a representative steady feedrate and compares it with observed operation time. The difference provides an estimate of potential time savings, which is further refined by analyzing air-cut segments and cornering behavior. Sharp directional changes are detected to account for motion constraints that may limit achievable gains.

Finally, all per-operation metrics, validated events, and time-saving estimates are aggregated and ordered according to the global priority score. The module generates interactive visualizations and rule-based textual recommendations that explicitly link suggested actions to the underlying measurements.

5.3.3 Decision Logic and Robustness

The implementation relies on conservative, explainable decision logic built on discrete counts, robust summary statistics, and simple geometric relationships. Thresholds are derived from robust measures and applied through interpretable rules that produce human-readable classifications such as *good usage*, *moderate usage*, or *ac-*

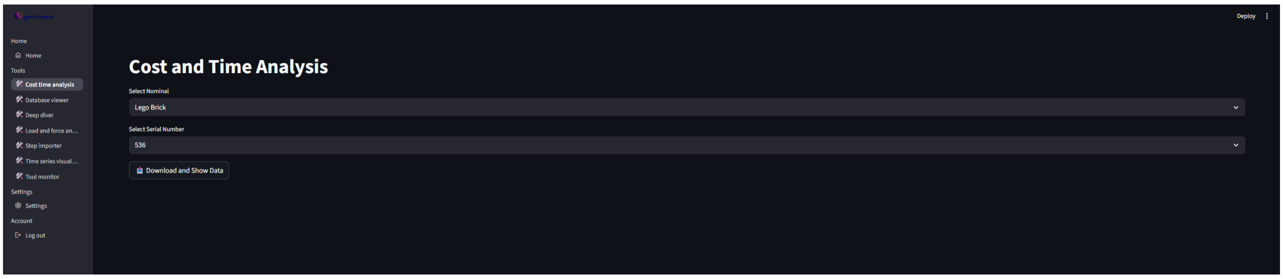
tion recommended. Throughout the pipeline, fallback policies ensure that missing or degenerate inputs yield stable, conservative results rather than runtime failures.

To support reproducibility and numerical stability, the module consistently normalizes temporal units, applies robust statistics to mitigate outliers, and maintains a modular structure that allows individual processing stages to be tested independently.

5.3.4 User Interface and Integration

The home page of the cost and time analysis module is shown in Figure 14. The interface integrates operation selection, prioritization, KPI visualization, and optimization feedback into a single workflow. Interactive plots and summaries allow users to explore both high-level time-saving potential and the detailed signal-level evidence supporting each recommendation.

Figure 14 – Cost and Time Module Home Page



Source: Author‘s archive

5.4 IMPLEMENTATION: LOAD AND FORCE ANALYSIS

5.4.1 Objective and Scope

The load and force analysis module processes per-operation time series of spindle load and cutting force measurements to identify, quantify, and contextualize transient load events associated with unfavorable cutting conditions. The objective is to convert raw sensor signals into validated and interpretable indicators of process instability, excessive loading, and potential quality or tool-integrity risks.

The module outputs:

- validated transient event lists with timestamps, amplitudes, and durations;
- aggregate indicators such as event rate and fraction of time under elevated load;
- spatial localization of detected events along the toolpath for visual inspection.

All analysis is performed in the time domain and prioritizes robustness, explainability, and traceability of decisions.

5.4.2 Processing Pipeline

Processing begins by ingesting per-operation telemetry and converting all channels (load, filtered load, force components, timestamps, and coordinates) into typed numeric arrays. Discrete sample counts are recorded for each channel and used as guard conditions; operations with insufficient data density are excluded and logged. These counts also parameterize later adaptive choices, including filter window lengths and minimum event durations.

5.4.3 Savitzky–Golay Baseline Estimation and Residual Formation

To isolate transient load phenomena while preserving the local shape of the signal, a Savitzky–Golay (SavGol) filter is applied to the raw load trace. The SavGol filter performs a local polynomial regression over a sliding window and is well suited for machining signals because it smooths high-frequency noise without significantly distorting peak amplitude or timing.

The filter window length is selected adaptively as an odd number proportional to the total sample count, with a lower bound to guarantee numerical stability. The polynomial order is fixed to a low degree to capture slow-varying load trends without fitting transient spikes.

Let $L_{\text{raw}}(t_i)$ denote the raw load signal at sample i. The estimated baseline is obtained using a Savitzky–Golay filter:

$$L_{\text{baseline}}(t_i) = \text{SavGol}\big(L_{\text{raw}}(t_i)\big), \tag{23}$$

and the residual signal used for transient detection is

$$r(t_i) = L_{\text{raw}}(t_i) - L_{\text{baseline}}(t_i). \tag{24}$$

This residual emphasizes abrupt deviations from nominal load behavior while suppressing low-frequency variations due to gradual engagement changes or toolpath geometry. The code implementation is given in Appendix B.

5.4.4 Robust Noise Scale Estimation

To enable adaptive and outlier-resistant thresholding, the dispersion of the residual signal is quantified using the median absolute deviation (MAD):

$$\text{MAD} = \text{median}\Big(\big|r(t_i) - \text{median}(r(t_i))\big|\Big), \tag{25}$$

where $r(t_i)$ is the residual load at sample i, and the median is taken over all samples of the operation. The MAD provides a robust estimate of background noise amplitude and remains stable in the presence of large transient events.

5.4.5 Candidate Event Detection and Acceptance

Candidate transient events are identified as local maxima in the residual signal. Each candidate at time t^* is accepted if

$$
\begin{aligned}
r(t^*) \geq H_{\text{min}} \quad &\text{and} \quad \text{prom}(t^*) \geq P_{\text{min}}, \\
\text{width}(t^*) \geq W_{\text{min}} \quad &\text{and} \quad \Delta t_{\text{neighbor}} \geq D_{\text{min}},
\end{aligned}
\tag{26}
$$

where H_{min} is an amplitude threshold defined as a multiple of MAD above the median residual, P_{min} is a minimum prominence derived from MAD and the signal dynamic range, W_{min} is the minimum temporal width expressed in seconds and converted to samples using the sampling interval Δt, and D_{min} is the minimum temporal separation between consecutive accepted events.

5.4.6 Artefact Rejection via Filter Consistency

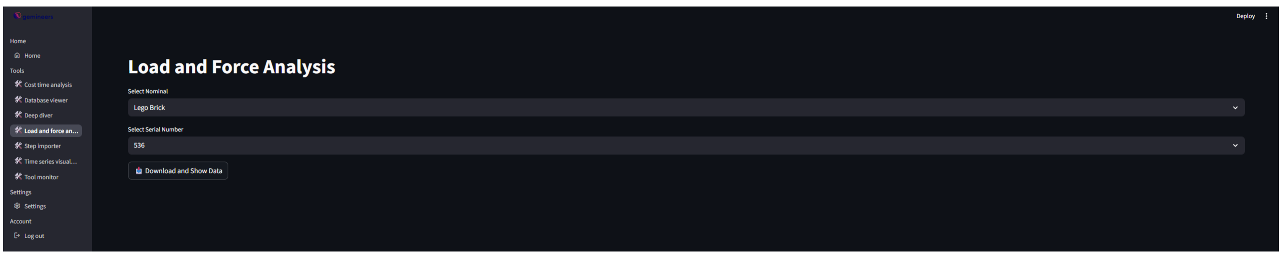
To reduce sensitivity to sensor artefacts and isolated spikes, candidate events are cross-validated against the Savitzky–Golay filtered trace. Genuine mechanical load events are expected to manifest coherently in both the raw and filtered signals. A candidate is rejected if the discrepancy between the two exceeds a conservative margin:

$$
L_{\text{raw}}(t^*) - L_{\text{baseline}}(t^*) > M_{\text{artifact}}, \tag{27}
$$

where M_{artifact} is defined as a fraction of the residual dynamic range or as a multiple of MAD.

The home page of the load and force analysis module is shown in Figure 15.

Figure 15 – Load and Force Module Home Page



Source: Author‘s archive

5.5 IMPLEMENTATION: TIME SERIES VISUALIZATION

The time series visualization module provides an interactive interface for exploring multi-signal, multi-operation datasets. Users can select operations from a global database, discover available telemetry signals, and choose between raw, cleaned, or combined traces. The visualization style can be configured as either time-aligned (*Series*) or index-aligned (*Overlap*), supporting chronological inspection and waveform comparison across operations.

Data is retrieved from a remote API and cached in session state to prevent redundant downloads. Available signals are enumerated across selected operations, filtered for valid entries, and presented to the user for selection.

Case A: Time-aligned display (Series style). When the visualization style is *Series*, the x-axis represents absolute or relative time. For raw data, the system extracts absolute timestamps (in milliseconds) and converts them to datetime objects:

$$t_i^{\text{abs}} = \text{datetime}(\text{ms}_i). \tag{28}$$

For cleaned data, the system retrieves relative timestamps and a start reference, computes absolute times and converts:

$$t_i^{\text{abs}} = \text{datetime}(\text{start\_ref} + \Delta t_i). \tag{29}$$

This ensures that traces from different operations or representations are aligned chronologically if they span overlapping time windows.

Case B: Index-aligned display (Overlap style). When the visualization style is *Overlap*, all traces are resampled to a common sample index, regardless of their original timestamps:

$$x_i = i, \quad i = 0, 1, 2, \ldots, N - 1, \tag{30}$$

where N is the number of samples in each trace. This mode is useful for comparing waveform shapes across different operations or time durations, abstracting away absolute timing differences.

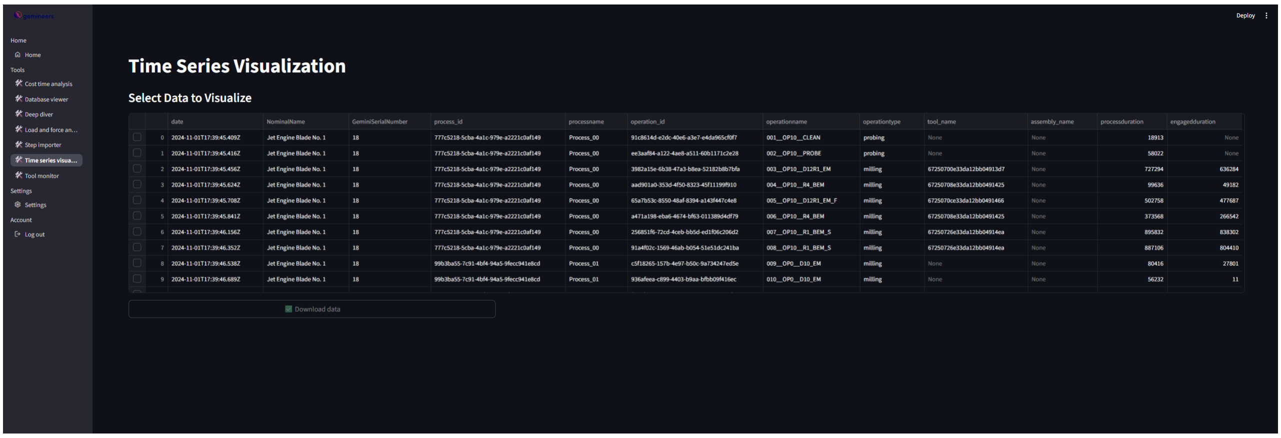
5.5.1 Visualization and Layout

Each operation is assigned a unique color from a deterministic palette to maintain trace consistency. Multi-panel subplots are created dynamically, one row per signal, with linked x-axes to synchronize zooming and panning. Trace objects carry x- and y-coordinates, line style (differentiating raw vs. cleaned), and legend entries. Axis labels combine signal names with units from metadata for self-documentation.

The module handles missing or empty data gracefully: operations or signals without valid entries are skipped, and raw and cleaned representations can be combined with deduplication.

The home page of the module is shown in Figure 16.

Figure 16 – Time Series Module Home Page



Source: Author‘s archive

5.6 IMPLEMENTATION: FREQUENCY ANALYSIS

5.6.1 User Workflow and Processing Pipeline

The coloquially called *Deep Diver* module provides an interactive environment for exploring signals in the time, frequency, and time-frequency domains. Users begin by either importing raw data files (ZIP archives containing preprocessed traces) or selecting a nominal process and serial instance from a remote database. Once the data is loaded, all available signals are enumerated and presented for selection. The user chooses a primary signal for analysis, after which the system computes summary statistics such as signal range, temporal extent, and sampling frequency. These values are stored in a `DataRange` object and serve as the basis for subsequent filtering and visualization.

Users can then interactively define time windows, frequency bands, and magnitude ranges, allowing focused inspection of specific signal segments or spectral regions of interest.

5.6.2 Figure Construction and Visualization Layout

The module visualizes the analysis using a multi-panel Plotly figure arranged in a 3×2 grid. The first column contains a 3D toolpath visualization spanning all rows, where each sample point is colored according to the magnitude of the selected signal, enabling spatial correlation between motion and signal behavior. The second column contains three vertically stacked 2D plots: the FFT-based power spectral density (PSD) in decibel scale, the raw time-series representation of the signal, and a short-time Fourier transform (STFT) spectrogram. This combined layout enables simultaneous inspection of spatial, temporal, and spectral characteristics.

5.6.3 Frequency Analysis and Spectral Filtering

The frequency-domain analysis begins with the computation of the discrete Fourier transform of the selected signal,

$$X(f_k) = \mathcal{F}\{x(t_n)\}, \tag{31}$$

where $x(t_n)$ denotes the sampled signal. The resulting magnitude spectrum is converted to decibel scale for visualization and interpretation.

To capture nonstationary spectral behavior, a short-time Fourier transform is computed using a sliding window approach. In the implementation, typical default parameters include a window duration of approximately 0.5 seconds, a hop fraction of 10% between consecutive windows, and a frequency oversampling factor of 3 to improve spectral resolution. These parameters define the window length, overlap, and FFT size used in the STFT computation.

User-selected frequency bands are applied through smooth bandpass filtering in the frequency domain. A soft-edged frequency mask $W(f)$ is constructed such that

$$X_{\text{filtered}}(f) = X(f) \cdot W(f), \tag{32}$$

where smooth transitions at the band edges are implemented using cubic smoothstep interpolation to reduce spectral leakage. The filtered time-domain signal is then reconstructed via inverse Fourier transform. To mitigate boundary artifacts introduced by filtering, padding corrections are applied near the start and end of the reconstructed signal.

5.6.4 Data Range Updates and Dynamic Scaling

After filtering and reconstruction, the system recomputes signal statistics to update visualization ranges. Linear magnitude limits for the time-series plot, spectral magnitude limits for the FFT, and color scale limits for the spectrogram are updated using quantile-based scaling,

$$\text{range}_{\text{low}} = Q_{0.01}, \quad \text{range}_{\text{high}} = \max(\cdot), \tag{33}$$

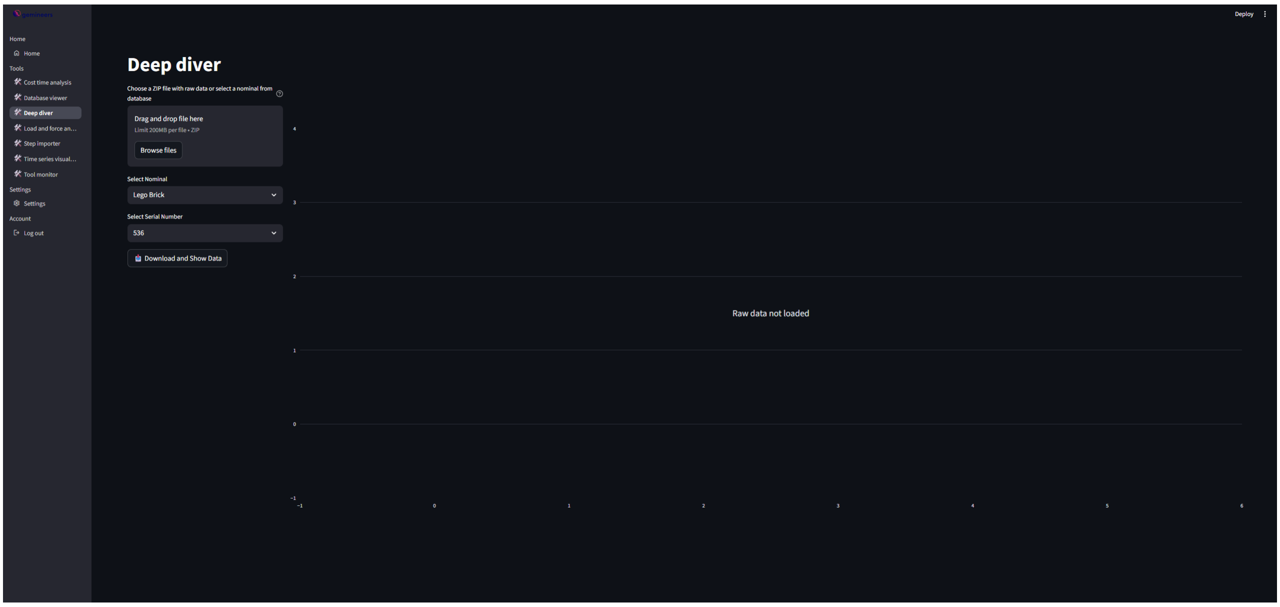
where $Q_{0.01}$ denotes the 1^{st} percentile of finite-valued data. This strategy ensures robustness to outliers while preserving visibility of low-amplitude but diagnostically relevant features. Updated ranges are stored in a filtered `DataRange` object, enabling iterative refinement of analysis parameters.

5.6.5 Interactive Updates and Session State Management

All user interactions—such as adjusting time windows, frequency bands, or magnitude sliders—trigger real-time updates of the visualization. The 3D toolpath, time-series plot, FFT spectrum, and spectrogram are refreshed consistently to reflect the current filtered state. Streamlit session state variables preserve the original and filtered datasets, computed ranges, and figure objects, enabling smooth interactive exploration without unnecessary recomputation or data reloading.

The home page of the module is illustrated in Figure 17.

Figure 17 – Frequency Module Home Page



Source: Author‘s archive

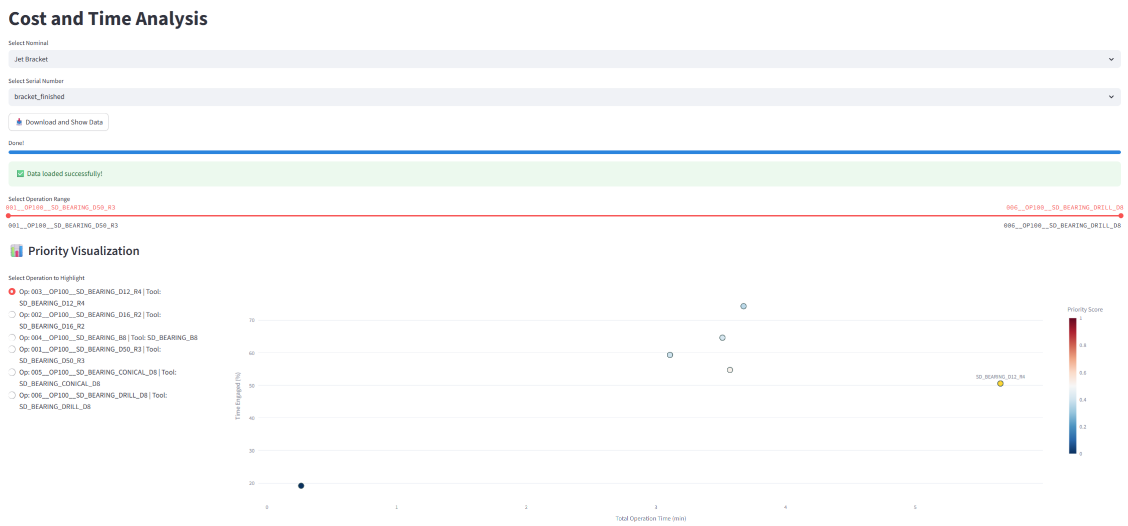
5.7 RESULTS

Here will be presented the final result of the application along with discussed and addressed examples of use cases of the modules on real clients situations.

5.7.1 Cost and Time Results

The analysis begins with the selection of the nominal, serial number, and the desired operations to be evaluated. Once the operations are selected, the module ranks them according to their optimization potential using a total operation time versus engaged time graph. This prioritization allows the analysis to focus first on the operations with the highest potential impact on overall process efficiency. Based on this ordering, the detailed analysis is performed. Different analysis pipelines are implemented for bullnose mills (end mills), ball mills, and drilling operations, reflecting the distinct cutting mechanics and optimization opportunities associated with each tool type.

Figure 18 – Operations Prioritized



Source: Author‘s archive

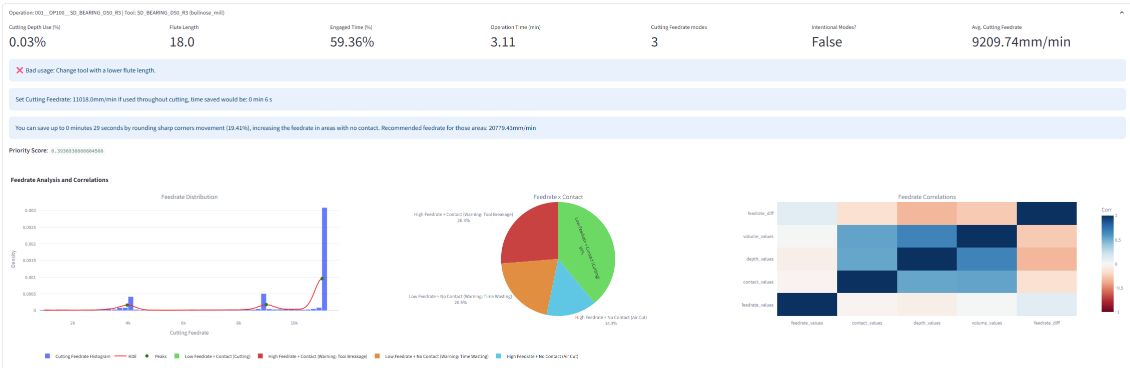
For end mill operations, the analysis interface initially presents a set of key performance indicators (KPIs) related to tool usage and process efficiency. These include the percentage of cutting flute length depth usage, defined as the percentage of cutting time during which more than 80% of the flute length is engaged; the total flute length of the tool; the percentage of engaged cutting time; the total operation time; the number of distinct cutting feedrate modes; whether these feedrate modes are intentional, determined through correlation between chip depth and feedrate; and the average cutting feedrate.

Based on these KPIs, the module generates optimization suggestions. One of the primary evaluations concerns chip depth utilization. If the chip depth usage is below 50%, the tool usage is classified as poor; between 50% and 80% as moderate; and above 80% as good. These classifications are used to suggest potential tool changes, such as selecting a tool with a shorter flute length when excessive flute length is unused, thereby reducing tool cost without compromising process capability.

Another optimization focuses on cutting feedrate consistency. The module infers the operator-defined set cutting feedrate as the median cutting feedrate and estimates the potential time savings if this feedrate were maintained throughout the entire cutting phase. Additionally, cornering movements are analyzed: the algorithm detects sharp corner transitions and estimates potential time savings by rounding these trajectories and increasing the feedrate in regions without tool–workpiece contact which the code is shown in Appendix A. In such air-cut regions, a suggested feedrate is computed based on the median air-cut feedrate.

The feedrate analysis and correlation section further explores process behavior. The cutting feedrate distribution is visualized using a Gaussian kernel density estimation (KDE) in a feedrate versus density plot. Alongside this, a pie chart presents the proportion of time spent in different operating conditions: low feedrate with cutting (ideal cutting condition), high feedrate without contact (air cutting), high feedrate with contact (potentially dangerous condition that may lead to tool breakage), and low feedrate without contact (inefficient air cutting and time waste). A correlation matrix between all relevant variables is also provided to support deeper process analysis. These configurations are shown in Figure 19.

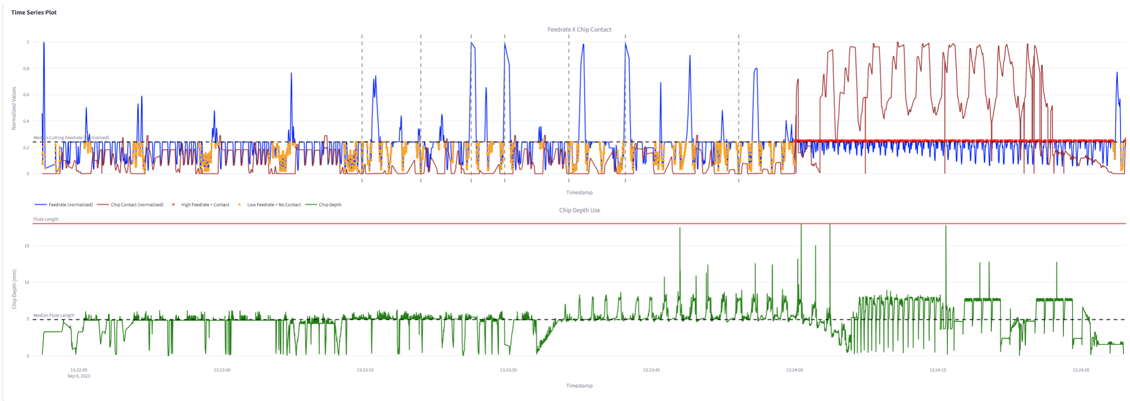
Figure 19 – First Part of End Mill Analysis



Source: Author‘s archive

The time-series visualization complements this analysis by displaying normalized feedrate and chip contact area over time. Warning markers highlight regions of high feedrate with contact and low feedrate without contact, which represent the main optimization targets, as well as cornering events. Beneath this plot, a chip depth usage graph summarizes flute utilization throughout the operation shown in Figure 20.

Figure 20 – Second Part of End Mill Analysis



Source: Author's archive

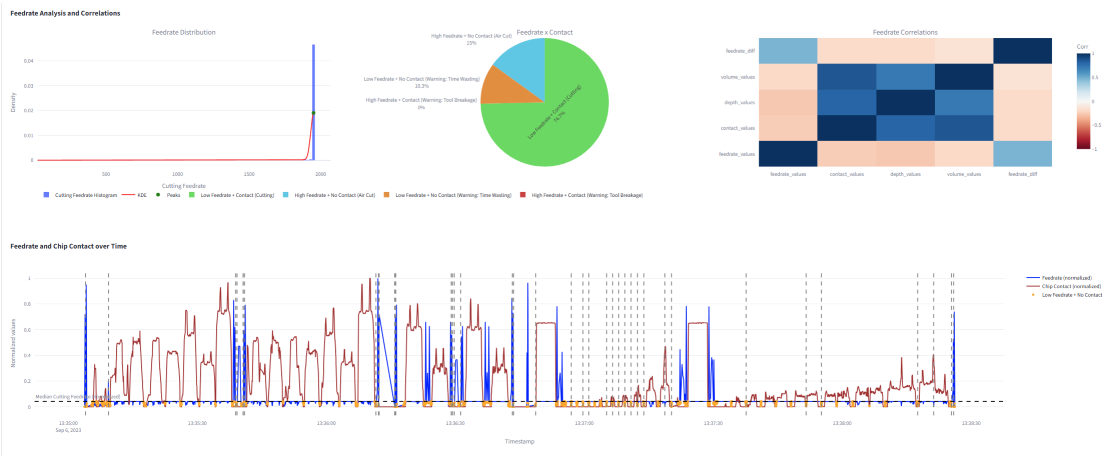
The ball mill analysis follows a similar structure, with adaptations to reflect the tool geometry and cutting mechanics. Instead of chip depth and flute length usage, the module evaluates the average angle of tool usage and the tool diameter. Optimization suggestions emphasize using a larger portion of the available cutting angle to improve tool utilization, which can be seen in Figure 21.

Figure 21 – First Part of Ball Mill Analysis

Source: Author's archive

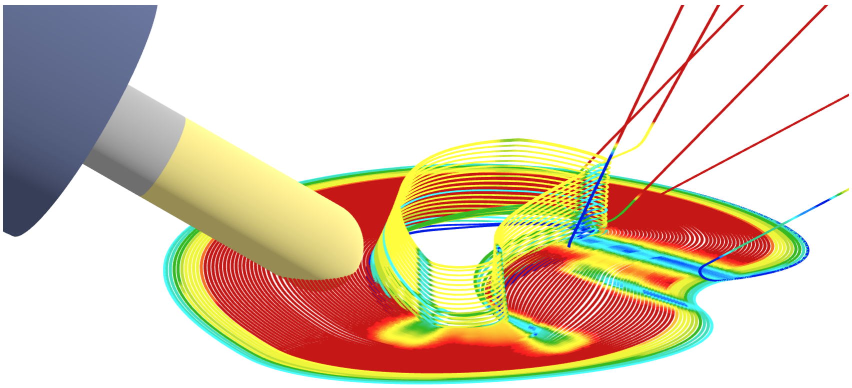
Before the feedrate analysis and correlation plots, the module presents the tool wear profile and the angular usage distribution as a function of removed volume. This includes the computation of a weighted average usage angle, providing insight into how the tool is loaded over time. The remaining analysis—feedrate distributions, operating condition classification, correlations, and time-series visualization—is analogous to the end mill case, except that the chip depth usage graph is not applicable and therefore omitted. This part is shown in Figure 22 and the reflection of the data seen in the Module is also shown in the toolpath inside the software, shown in Figure 23.

Figure 22 – Second Part of Ball Mill Analysis



Source: Author‘s archive

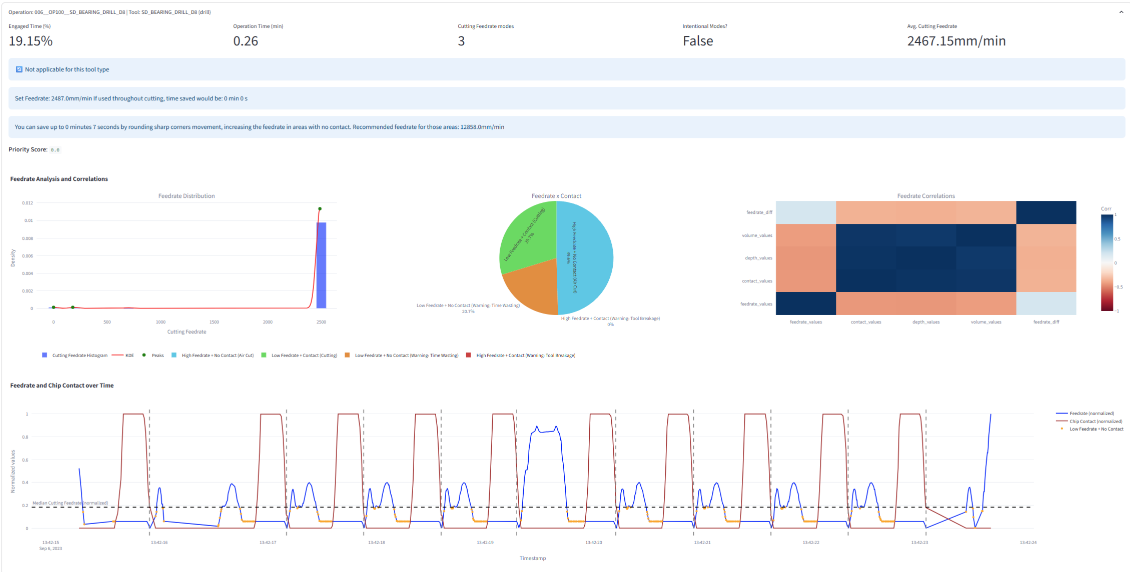
Figure 23 – Tool angle Seen in the Software Toolpath



Source: Author‘s archive

For drilling operations, neither flute length usage nor angular usage analysis is applicable. Consequently, the cost and time module focuses primarily on feedrate behavior and its associated optimization opportunities, using the same feedrate classification and time-series visualization concepts described previously as shown in Figure 24.

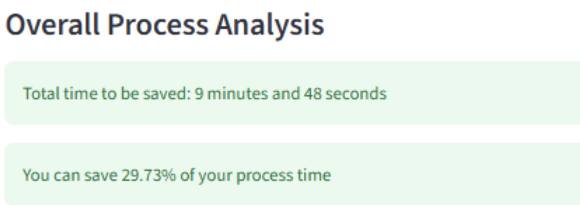
Figure 24 – Drilling Analysis



Source: Author‘s archive

After processing all operations associated with a given serial number, the module provides an aggregated estimate of the total time that could be saved, along with the corresponding percentage reduction in process time shown in Figure 25. While the specific numerical results are discussed in detail in the conclusion chapter, these estimates demonstrate the potential for significant efficiency improvements at the part level. When applied to batch production of identical parts, the accumulated time savings become even more substantial.

Figure 25 – Time Overall Results



Source: Author‘s archive

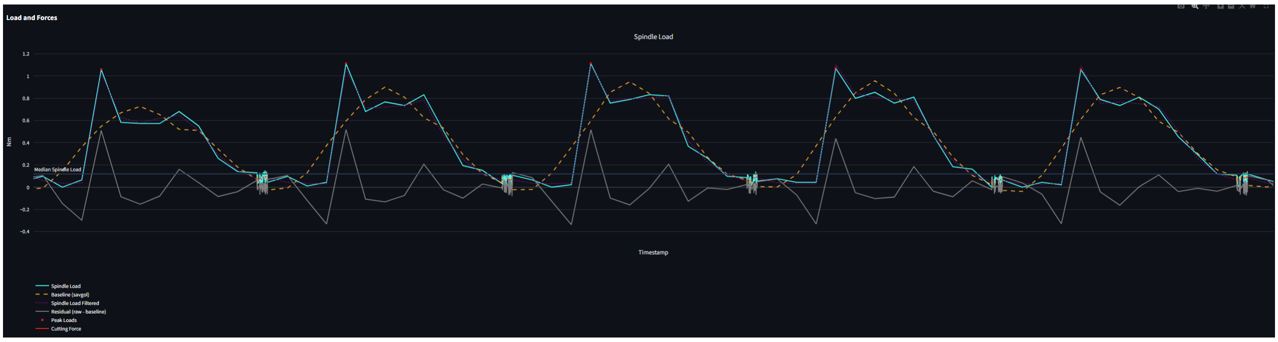
From a cost perspective, improved utilization of end mills and ball mills directly reduces tooling expenses by avoiding premature tool replacement and preventing the use of oversized or unnecessarily expensive tools. For example, tools with longer flute lengths or larger diameters typically incur higher costs; selecting tools that more closely match the actual process requirements can therefore yield meaningful cost reductions.

Together, these results highlight the effectiveness of the cost and time module in identifying both productivity and economic optimization opportunities across machining processes.

5.7.2 Load and Force Results

The use case for this module is applied to a real client case involving reamer tool breakage during drilling operations. The objective of this analysis is to identify abnormal load behaviors that may indicate tool degradation or imminent failure.

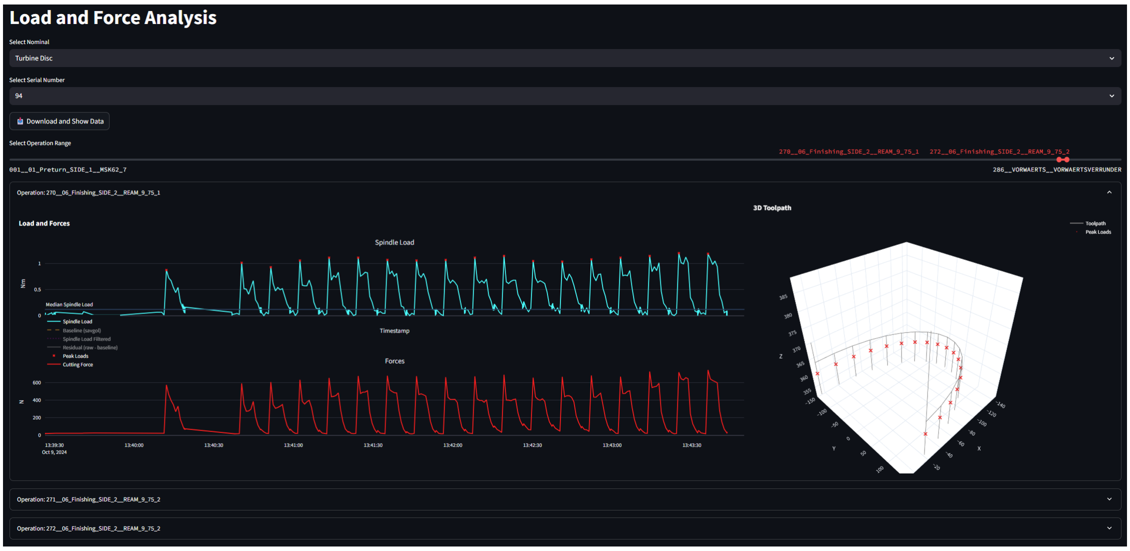
Figure 26 – SavGol Filter Used on the Load



Source: Author‘s archive

Figure 26 illustrates the application of the Savitzky–Golay (SavGol) filter to the spindle load signal for a single drilling operation (Savitzky; Golay, 1964). The figure shows the raw spindle load, the filtered signal representing the estimated baseline behavior, and the resulting residual signal obtained by subtracting the baseline from the raw data. When the residual exceeds a predefined threshold, a peak is detected and flagged by the algorithm. These peaks do not necessarily indicate an immediate failure, but rather represent warnings of load values that deviate from the expected pattern for the operation. This approach allows the system to highlight transient overload conditions that may warrant further investigation.

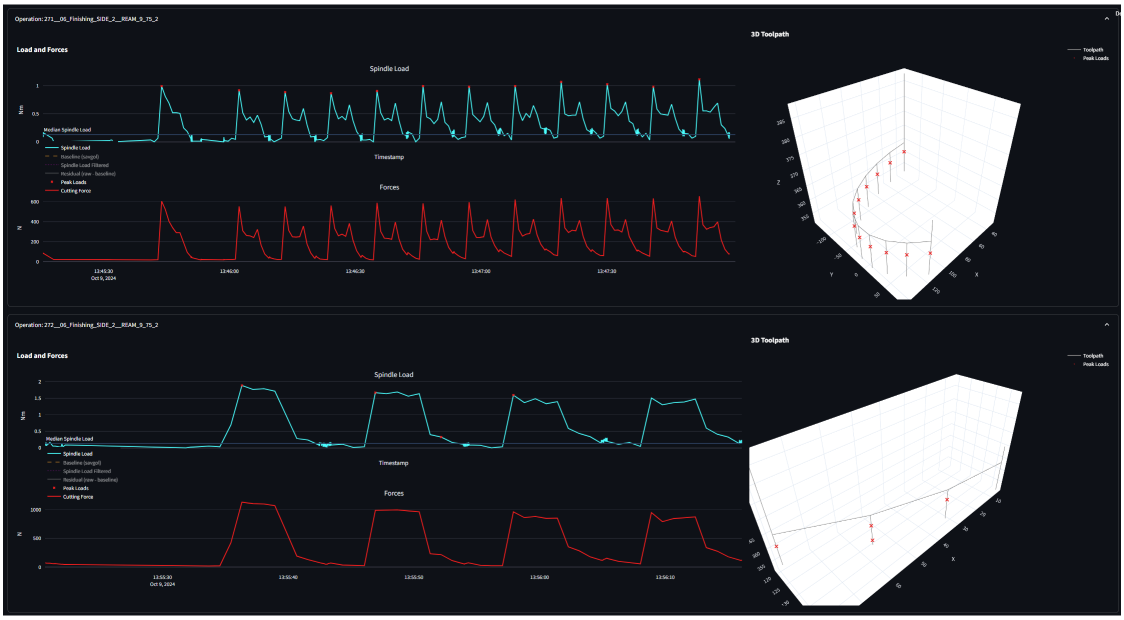
Figure 27 – Load and Force Analysis of One Operation



Source: Author‘s archive

Figure 27 presents the load and force analysis for the first drilling operation associated with the selected nominal and serial number. In this operation, several peaks are detected in the spindle load signal; however, their magnitude and distribution are consistent with normal process variability. Both the load and force signals exhibit stable behavior, indicating that the operation can be classified as normal despite the presence of detected peaks. Here is also shown the selection of geminis, serial numbers and operations to be analyzed.

Figure 28 – Load and Force Analysis of Two Operations



Source: Author‘s archive

Figure 28 shows the analysis of the subsequent two drilling operations. The second operation continues to exhibit a load pattern similar to the first, with no significant anomalies. In contrast, the third operation displays a markedly different spindle load behavior. The deviation from the expected pattern is sufficiently large that it interferes with the peak detection configuration, leading to the identification of an incorrect or spurious peak. This behavior is a strong indicator of an abnormal process condition and suggests the onset of tool failure, which in this case culminates in reamer breakage. A deeper investigation of this event is carried out in the frequency-domain analysis presented in the following section.

Throughout all operations, the force signals remain within acceptable limits and do not show significant anomalies. In addition to the temporal analysis, the corresponding 3D tool path and the spatial distribution of detected peaks are also visualized, complementing the 2D load and force plots and providing further insight into the process behavior.

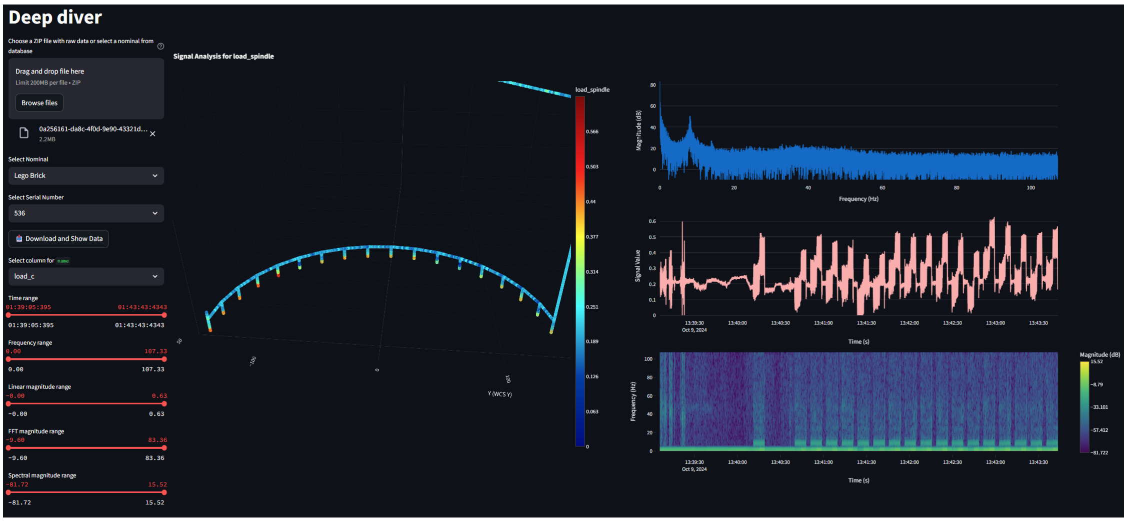
5.7.3 Frequency Results

Here will be presented the frequency-domain analysis of the three drilling operations associated with the reamer breakage case. This module supports two data ingestion methods: direct retrieval from the database and manual insertion of raw signal data. In the present analysis, raw data were used as input, allowing full control over signal selection and preprocessing.

Once the data are loaded, the interface enables the visualization of different signals and representations, including the time-domain signal, frequency spectrum, FFT magnitude, spectral magnitude, and linear magnitude filtering. Additionally, the 3D tool path is displayed and color-coded according to the selected signal, providing spatial context to the frequency behavior. The final column of the interface consolidates the time series, FFT, and Short-Time Fourier Transform (STFT) plots, allowing a comprehensive assessment of both stationary and non-stationary frequency components.

Following the load anomalies identified in the time-domain analysis, the spindle load signal on the C axis was examined in the frequency domain for all three operations. As shown in Figure 29, the first operation exhibits a stable frequency response, with the FFT and STFT dominated by a single component at approximately 9 Hz. This frequency corresponds to the rotational frequency of the tool teeth and is characteristic of normal cutting conditions, with no evidence of abnormal vibrations.

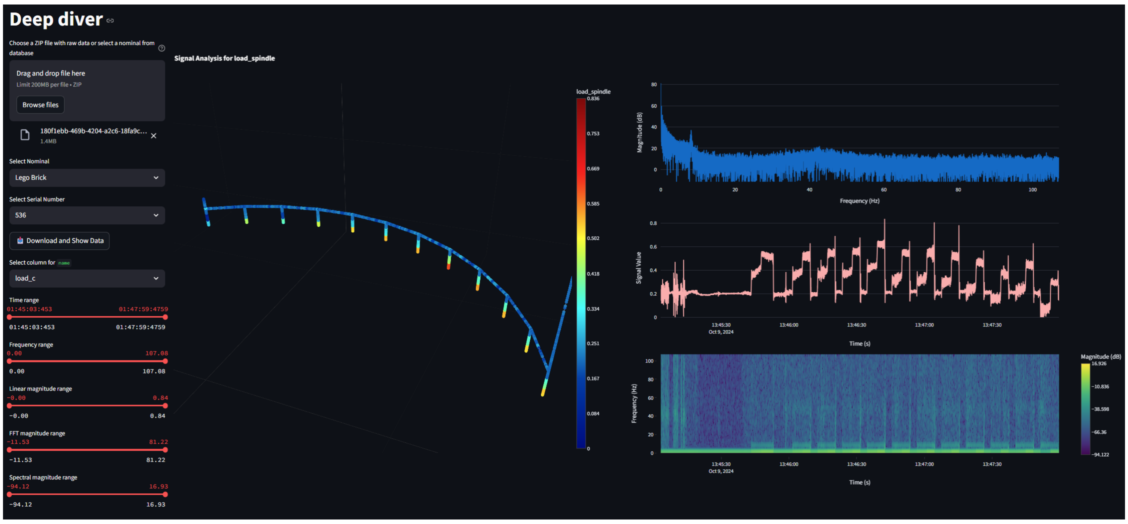
Figure 29 – Frequency Analysis of First Operation



Source: Author‘s archive

Figure 30 shows the frequency analysis of the second operation. The overall spectral content remains similar to that of the first operation, with no significant additional frequency components or broadband energy increase. Both the FFT and STFT indicate a stable process, corroborating the conclusions drawn from the load analysis.

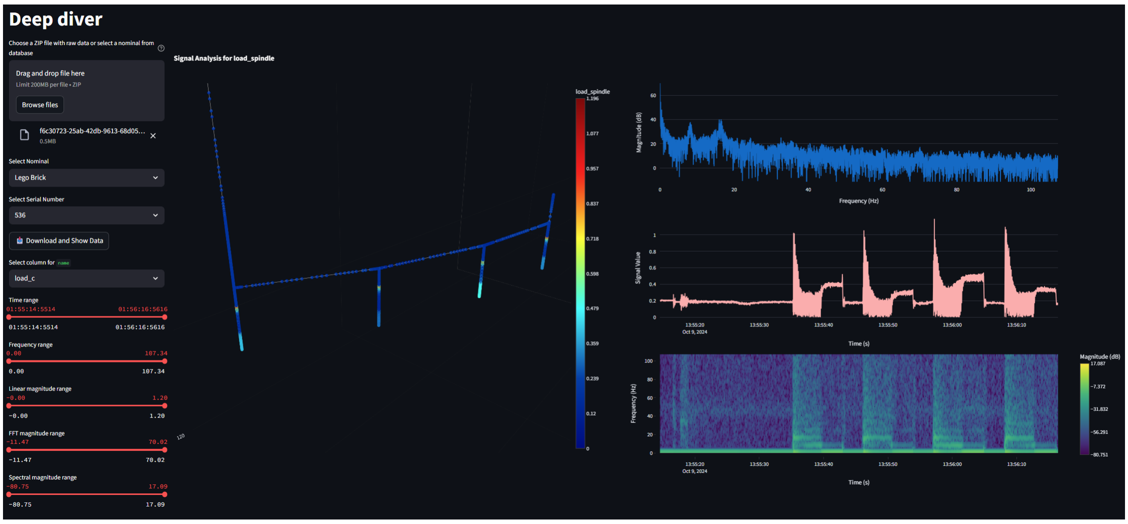
Figure 30 – Frequency Analysis of Second Operation



Source: Author‘s archive

In contrast, Figure 31 reveals a clear change in the frequency behavior of the third operation. Along with the observed increase in spindle load, a new dominant frequency component emerges at approximately 20 Hz. This frequency is not harmonically related to the tool rotational frequency and is indicative of self-excited vibrations, commonly referred to as chatter. The presence of this component is further confirmed by the STFT, which shows sustained energy at this frequency over time. Consequently, the time-domain signal exhibits a significantly higher noise level compared to the previous operations. These results strongly suggest unstable cutting conditions and provide a frequency-domain explanation for the abnormal load patterns observed prior to tool breakage.

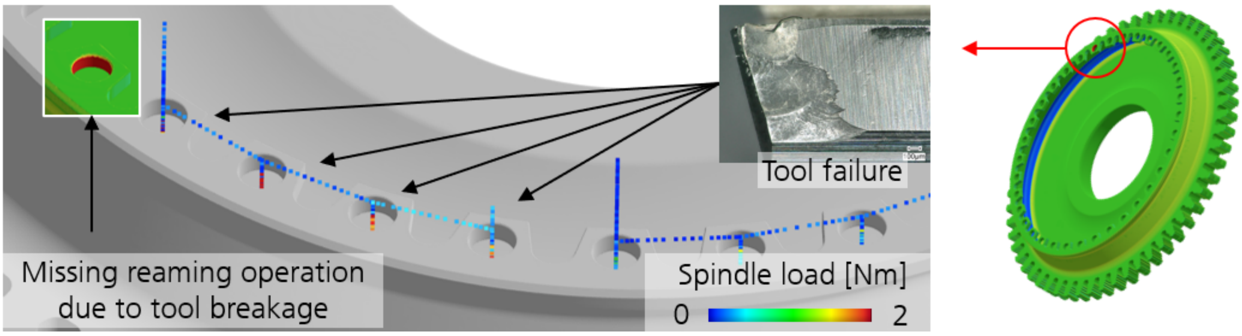
Figure 31 – Frequency Analysis of Third Operation



Source: Author's archive

Later checking shows that the reamer broke, and a operation was missed because of it, which also can be seen in the digital twin in the Figure

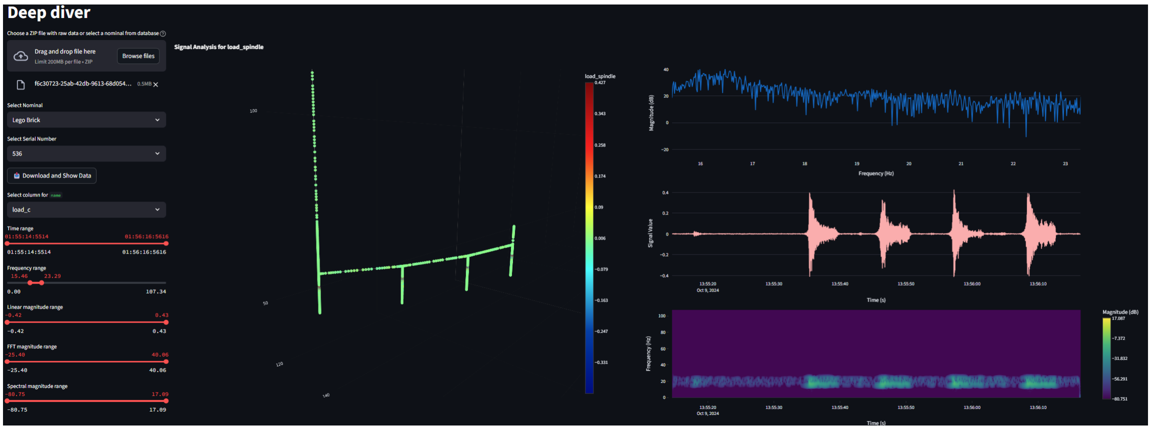
Figure 32 – Tool Breakage



Source: Company's archive

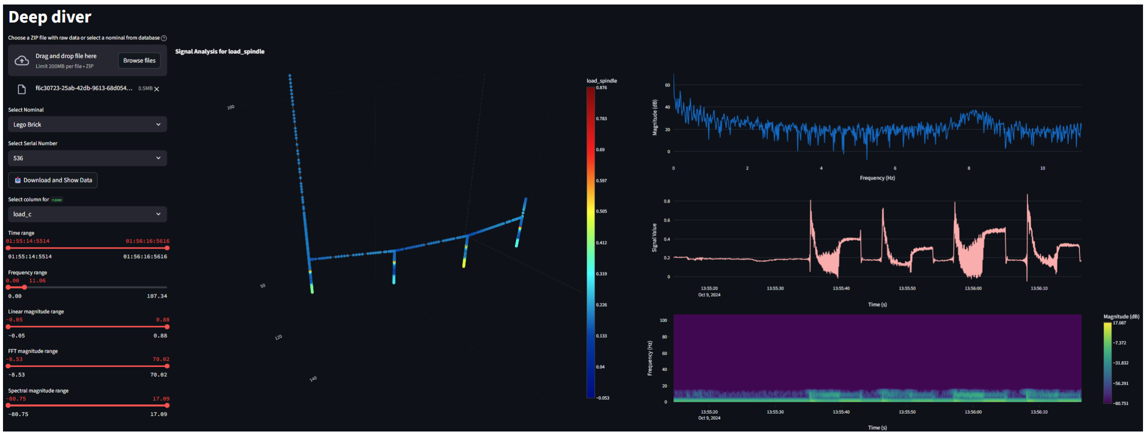
Using the filters, it's possible to isolate the chattering frequency thorugh a dynamic band pass filter and isolate the desired frequencies. Figure 33 shows only the chattering frequency behaviour 20Hz, and sequentially, Figure 34 shows the signal without it.

Figure 33 – Chatter Frequency Filtered



Source: Author‘s archive

Figure 34 – Signal without Chattering Frequency

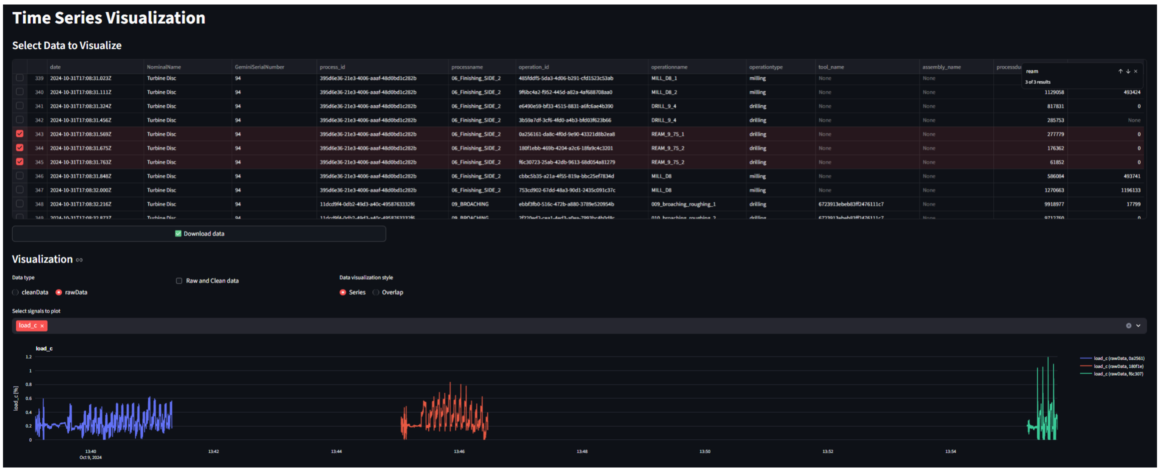


Source: Author‘s archive

5.7.4 Time Series Results

Figures 35 and 36 provide an alternative visualization of the spindle load on the C axis for the three drilling operations associated with the reamer breakage case. In Figure 35, the load signals are plotted against time, allowing a direct comparison of their temporal evolution. This representation highlights the progressive increase in load magnitude across the operations, with the third operation exhibiting a noticeably higher and more irregular load profile.

Figure 35 – Time Series of Load C in the 3 Drilling Operations



Source: Author‘s archive

Figure 36 – Time Series Overlapped of Load C in the 3 Drilling Operations

Source: Author‘s archive

Figure 36 shows the same three load signals overlaid in a single plot. By aligning the time series, differences in load behavior become more evident, clearly emphasizing the load increase and variability in the third operation compared to the first two. This visualization reinforces the conclusions drawn from both the time-domain peak analysis and the frequency-domain results, confirming the abnormal behavior preceding tool failure.

Figures 37 and 38 present a second application of the time-series module, this time focusing on milling operations performed on the same product but using different materials. Material 1 (Mat. 1) corresponds to an Alloy 718 condition that has been solution annealed and precipitation hardened, resulting in a high hardness of 422 HBW

(Brinell hardness measured with a tungsten carbide indenter) (Yin, 2026). Material 2 (Mat. 2) is solution annealed without precipitation hardening, leading to a significantly lower hardness of 199 HBW.

Figure 37 – Spindle Load Signal Overlapped of Two Identical Operations in Different Materials

Source: Author‘s archive

Figure 38 – Spindle Load Signal Overlapped of Two Identical Operations in Different Materials

Source: Author‘s archive

Despite the milling operations being geometrically and parametrically identical, the difference in material properties has a direct impact on the machining response.

When the spindle load time series for both materials are overlaid, a clear increase in load is observed for Mat. 1 compared to Mat. 2. This behavior is consistent with the higher cutting forces and increased tool–workpiece interaction associated with the harder material.

This module is a practical and efficient tool for the visualization and comparison of machining signals. Its implementation allows the user to access all operations available in the database, enabling the selection and download of specific datasets for analysis. For each operation, the user can choose whether to work with raw data, cleaned data, or both simultaneously, providing flexibility in assessing signal quality and preprocessing effects.

In addition, the module supports the selection of one or multiple signals to be displayed at the same time, facilitating direct comparison between different process variables. The visualization method can also be configured, allowing the signals to be plotted either along their original timestamps or overlaid in a normalized time axis. This versatility makes the module particularly useful for quick exploratory analysis, comparative studies between operations, and the identification of trends or anomalies across different machining conditions.

6 CONCLUSION

6.1 SUMMARY OF THE PROBLEM AND PROPOSED SOLUTION

Modern machining processes, especially in high-value industrial sectors such as automotive and aerospace manufacturing, operate under strict constraints of cost, time, and quality. The increasing complexity of parts, the high cost of raw materials, tools, and machinery, and the reduced tolerance for errors make inefficiencies and undetected process issues particularly expensive. Despite the large amount of data generated during machining operations, this information is often underutilized, limiting the ability to detect anomalies, optimize parameters, and support informed decision-making.

The problem addressed in this work lies in the lack of integrated, accessible, and data-driven tools capable of transforming raw machining data into actionable insights. Without proper analysis, issues such as excessive loads, abnormal vibrations, tool wear, or inefficient cycle times may remain unnoticed until they result in tool breakage, scrap, or unplanned downtime, directly impacting productivity and cost.

To address this challenge, this dissertation proposes the development of an extension to an existing software framework that applies data analysis algorithms to machining process data. The proposed solution integrates mathematical and statistical techniques, including filtering, correlation analysis, frequency-domain analysis using Fourier transforms, and time-series analysis, to extract meaningful indicators related to cost, time, and quality. These indicators enable the detection of anomalies, the evaluation of process performance, and the identification of optimization opportunities.

The solution is implemented using Python and its data analysis ecosystem, ensuring flexibility, scalability, and compatibility with industrial environments. An interactive frontend developed with Streamlit provides a user-friendly interface, allowing engineers and operators to easily interpret results without requiring advanced knowledge of data science. By combining robust analytical methods with practical usability, the proposed approach offers a structured and effective way to support data-driven optimization in machining processes.

6.2 RESULTS AND IMPACT

This work presented a multi-module analytical framework for machining process monitoring and optimization, integrating time-domain, frequency-domain, time-series comparison, and cost-and-time analysis. Although the proposed system was not deployed in real-time production environments, the results obtained from historical industrial data allow for a quantitative and qualitative assessment of its potential impact on process efficiency, tooling cost, and product quality. The following discussion summarizes the expected benefits under realistic hypothetical scenarios.

6.2.1 Impact on Product Quality and Process Stability

From a quality perspective, the load, frequency, and time-series analyses demonstrated a strong capability to detect abnormal machining behavior before catastrophic failure. In the reamer breakage case study, deviations in spindle load were first identified through residual analysis and peak detection. While early peaks alone did not necessarily indicate failure, the subsequent frequency-domain analysis revealed the emergence of a non-harmonic frequency component at approximately 20 Hz, characteristic of self-excited vibrations (chatter). This behavior was further corroborated by increased noise levels in the time-domain signal.

In a hypothetical real-time deployment, the early detection of such abnormal patterns could allow operators or automated systems to intervene before tool breakage or part scrap occurs. Preventing a single tool breakage event can result in savings related to tool replacement, machine downtime, and rejected parts. In addition, avoiding chatter improves surface finish, dimensional accuracy, and overall product consistency. Even conservative estimates suggest that early detection of unstable cutting conditions could reduce scrap rates and rework, leading to a measurable improvement in product quality and process reliability.

6.2.2 Impact on Tooling Cost

The cost and time module revealed several opportunities for reducing tooling costs through improved tool utilization. In end-mill operations, the analysis showed that a significant portion of cutting time may occur with less than optimal flute length engagement. In hypothetical scenarios where chip depth usage remains below 50%, replacing a tool with a large flute length by one with a shorter flute length—while maintaining identical cutting capability—could lead to direct cost savings. Tools with shorter flute lengths and smaller diameters are typically less expensive, and selecting them appropriately avoids over-specification.

Similarly, for ball mill operations, the analysis of angle usage and wear distribution highlighted cases where only a limited portion of the tool's cutting angle was effectively used. Encouraging the use of a larger portion of the available cutting angle can delay tool wear concentration and extend tool life. Hypothetically, even a modest increase in tool life—such as avoiding one premature tool change per batch—can yield meaningful cost reductions when scaled across high-volume production.

Additionally, identifying hazardous cutting conditions, such as high feedrate with tool–workpiece contact, reduces the likelihood of sudden tool breakage. Preventing such events avoids not only the cost of the tool itself but also secondary costs related to machine stoppages and potential damage to fixtures or workpieces.

6.2.3 Impact on Machining Time and Productivity

Time savings constitute one of the most significant potential impacts identified by this work. Across multiple operations, the analysis estimated time losses associated with low feedrate air cutting, inefficient cornering movements, and inconsistent cutting feedrates. By inferring the operator-defined set feedrate and estimating the potential savings if this feedrate were applied consistently during cutting, the module provides a quantitative estimate of recoverable machining time.

Furthermore, the detection of sharp cornering movements and excessive deceleration in non-cutting regions suggests additional optimization potential through trajectory smoothing and feedrate increases in air-cut zones. While the exact numerical values depend on the specific operation and part geometry, even small per-operation savings can accumulate significantly. In batch production scenarios, time savings scale linearly with the number of parts produced, potentially resulting in substantial reductions in total machine occupation time.

Reduced machining time directly translates into increased machine availability, higher throughput, and lower cost per part. In hypothetical production environments, these gains could enable manufacturers to meet higher demand without additional capital investment in machinery.

6.2.4 Overall Industrial Impact

Taken together, the results indicate that the proposed framework has the potential to positively impact manufacturing processes across three key dimensions: quality, cost, and time. Quality improvements arise from earlier detection of unstable cutting conditions and tool degradation, reducing scrap and rework. Cost reductions stem from better tool selection, extended tool life, and avoidance of catastrophic failures. Productivity gains are achieved through feedrate optimization, reduction of unnecessary air cutting, and improved motion efficiency.

Although the results presented in this work are based on offline analysis and hypothetical scenarios, they are grounded in real industrial data and realistic machining constraints. As such, they provide a strong indication that deploying this system in real-time or near-real-time environments could yield measurable and economically relevant benefits. Future work should focus on real-time integration, closed-loop control strategies, and long-term validation across multiple products and production lines to quantify these impacts with greater precision.

6.3 LIMITATIONS AND CHALLENGES

Despite the positive results achieved in the development of the proposed framework, this work faced several limitations and challenges that must be acknowledged.

One of the main challenges is related to algorithm parameterization. Many of the applied mathematical and statistical methods, such as filters, anomaly detection techniques, and frequency-domain analyses, require careful tuning of parameters. These parameters are often dependent on the machining process, tool type, material, and operating conditions, which limits the direct generalization of the solution across different machines or production environments without additional calibration.

Another relevant challenge concerns the testing and validation of algorithms. Although the framework was designed to be modular and scalable, the availability and diversity of real machining data can restrict comprehensive validation. In industrial scenarios, datasets may be limited, noisy, or incomplete, which affects the robustness of testing and may reduce confidence in the general applicability of the results. Furthermore, validating anomaly detection algorithms is inherently difficult, as true failure events such as tool breakage are relatively rare.

From a software engineering perspective, API integration posed additional complexity. Ensuring consistent communication between different system components and external data sources requires careful handling of data formats, synchronization, and error management. Any changes in upstream data structures or APIs may demand adjustments in the framework, increasing maintenance effort.

Performance and code optimization also represent an important limitation. As machining datasets can be large and high-frequency, computational efficiency becomes critical, especially when applying frequency analysis or time-series processing. While Python offers extensive analytical capabilities, achieving real-time or near-real-time performance may require further optimization, parallelization, or migration of specific routines to more performant implementations.

Additionally, the intersection between file structures and code logic introduces challenges related to scalability and robustness. The framework relies on consistent data organization and naming conventions for correct processing. Variations in file formats, missing data, or inconsistent metadata can lead to processing errors and require additional validation layers.

Finally, a broader limitation lies in the scope of the study, which focuses primarily on data analysis and visualization rather than closed-loop control or automatic parameter adjustment. While the proposed system supports decision-making and process optimization, it does not autonomously modify machining parameters, leaving final decisions to the user.

Overall, these limitations highlight opportunities for future improvements, such as adaptive parameter tuning, expanded validation with diverse datasets, improved performance optimization, and deeper integration with industrial control systems. Addressing these challenges would further enhance the robustness and industrial applicability of the proposed solution.

6.4 FUTURE WORK

The limitations identified in this work also indicate several promising directions for future development. One natural extension of the proposed framework is its application to a broader range of machining processes beyond the scenarios considered in this study. Adapting the analytical structure to processes such as turning, drilling, or grinding would increase the generality and industrial relevance of the solution, while also enabling comparative analyses across different manufacturing operations.

A further direction for future work involves expanding the scope of testing and validation by gathering larger and more diverse datasets. The availability of extensive machining data would enable more robust comparisons across tools, materials, and operating conditions, facilitating the identification and patternization of recurring behaviors within the processes. Based on these patterns, data-driven models could be developed to provide deeper insights into process dynamics and performance. In parallel, closer collaboration with end users and industrial partners would allow continuous refinement of the framework, ensuring that the analytical outputs and indicators are closely aligned with real-world use cases and customer needs.

Future work may also focus on improving real-time data handling and system scalability. Enhanced integration with industrial data sources, including direct machine interfaces and cloud-based analytics platforms, would allow continuous monitoring and more responsive analysis. Such developments could support near real-time feedback and facilitate deployment in distributed manufacturing environments.

Finally, expanding the visualization and user interaction capabilities represents a valuable opportunity for improvement. More advanced dashboards, customizable views, and interactive exploration of indicators could further simplify data interpretation and enhance user engagement. These enhancements would strengthen the framework's role as a decision-support tool, bridging the gap between complex data analysis and practical industrial application.

6.5 FINAL REMARKS

This work reinforces the role of data-driven approaches as a practical pathway to improve machining performance under the competing constraints of time, cost, and quality. By turning heterogeneous process signals into interpretable indicators and interactive visual evidence, the proposed framework helps reduce the gap between raw telemetry and engineering decision-making.

Beyond the specific case studies discussed, the main contribution lies in providing a modular and extensible analysis structure that can be adapted to different tools, operations, and datasets, while keeping the outputs explainable and actionable. The combination of robust signal processing, well-defined KPIs, and an accessible visualization layer supports a more systematic optimization workflow, in which inefficiencies and abnormal behaviors can be identified earlier and investigated with greater confidence.

Ultimately, this project highlights that meaningful improvements in manufacturing do not necessarily require fully autonomous control systems; significant value can already be achieved by enabling engineers to see, quantify, and prioritize opportunities for improvement using the data that is already available in modern machining environments.

REFERENCES

ALPAKJIAN, Serope; SCHMID, Steven R. **Manufacturing Processes for Engineering Materials**. [*S. l.*]: Prentice Hall, 2001.

BOOTHROYD, Geoffrey; KNIGHT, Winston A. **Fundamentals of Machining and Machine Tools**. [*S. l.*]: Marcel Dekker, 1989.

BOZMAN, Nick. **Docker for Developers: Develop and Deploy Modern Applications with Docker**. [*S. l.*]: Packt Publishing, 2024. ISBN 978-1-80355-789-5.

CERONE, L.; DE CHIFFRE, L.; HANSEN, H. N. Tool Wear Monitoring with Artificial Intelligence Methods: A Review. **Journal of Manufacturing and Materials Processing**, v. 7, n. 4, p. 129, 2023.

CNCLATHING. **What is Profile Milling: Profile Milling Process, Tools & Operation Tips**. Available from: `https://www.cnclathing.com/guide/what-is-profile-milling-profile-milling-process-tools-operation-tips-cnclathing`. Acesso em: 28 jan. 2026.

DRUMOND, Claire. **What is scrum and how to get started**. Available from: `https://www.atlassian.com/agile/scrum`. Acesso em: 28 jan. 2026.

ECOREPRAP. **CNC Drilling Machine: Everything You Need to Know**. Available from: `https://ecoreprap.com/cnc-machining/cnc-drilling-machine/`. Acesso em: 28 jan. 2026.

FORTUNATO, Nicolas. **Interactive Data Visualization with Python: Create Responsive and Dynamic Data Visualizations with Plotly**. [*S. l.*]: Packt Publishing, 2023. ISBN 978-1-80355-182-3.

FRAUNHOFER IPT. **Fine machining and optics – Passion for precision**. Available from: `https://www.ipt.fraunhofer.de/en/Profile/departments/fine-machining-optics.html`. Acesso em: 28 jan. 2026.

FRAUNHOFER IPT. **Fraunhofer Institute of Production Technology**. Available from: `https://www.ipt.fraunhofer.de/en/Profile.html`. Acesso em: 28 jan. 2026.

FRAUNHOFER IPT. **High performance cutting – Modern machining in the digital age**. Available from: `https://www.ipt.fraunhofer.de/en/Profile/departments/high-performance-cutting.html`. Acesso em: 28 jan. 2026.

GARUDA3D. **All About STL File Format**. Available from: `https://garuda3d.com/all-about-stl-file-format/`. Acesso em: 28 jan. 2026.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital Image Processing**. [*S. l.*]: Prentice Hall, 2008. ISBN 978-0-13-168728-8.

GOUY, Mathieu. **The JSON Handbook**. [*S. l.*]: O'Reilly Media, 2006. ISBN 978-0-596-52926-7.

KENDALL, Maurice G. The Treatment of Ties in Ranking Problems. **Biometrika**, v. 51, n. 3/4, p. 307–320, 1962. DOI: `10.1093/biomet/51.3-4.307`.

KISTLER. **Cutting forces in turning operations**. Available from: `https://www.kistler.com/INT/en/cutting-forces-in-turning-operations/C00000103`. Acesso em: 28 jan. 2026.

LUTZ, Mark. **Learning Python**. [*S. l.*]: O'Reilly Media, 2013. ISBN 978-0-596-15806-4.

LYONS, Richard G. **Understanding Digital Signal Processing**. [*S. l.*]: Pearson Education, 2011.

MICROSOFT. **Visual Studio Code**. Available from: `https://visualstudio.microsoft.com/pt-br/`. Acesso em: 28 jan. 2026.

MONGODB. **Relational vs. Non-Relational Databases**. Available from: `https://www.mongodb.com/compare/relational-vs-non-relational-databases`. Acesso em: 28 jan. 2026.

NTI AUDIO. **Fast Fourier Transformation FFT - Basics**. Available from: `https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft`. Acesso em: 28 jan. 2026.

O'GORMAN, Liz. **Decomposing Fourier transforms – an introduction to time-frequency decomposition**. Available from: `https://dibsmethodsmeetings.github.io/fourier-transforms/`. Acesso em: 28 jan. 2026.

OPPENHEIM, Alan V.; WILLSKY ALAN S. WITH NAWAB, S. Hamid. **Signals and Systems**. [*S. l.*]: Prentice Hall, 1999. ISBN 978-0-13-814757-0.

RICHARDSON, Leonard; AMUNDSEN, Mike; RUBY, Sam. **RESTful Web APIs**. [*S. l.*]: O'Reilly Media, 2020. ISBN 978-1-492-00741-3.

RITOU, M.; LEE, K. S.; HWANG, B. J. Monitoring and processing signal applied in machining processes – A review. **Measurement**, v. 58, p. 73–86, 2014.

RUBIO, Eduardo; JÁUREGUI-CORREA, Juan Carlos. Time–Frequency Approach for Cutting Tool Power Signal Separation in Face Milling Operations. **Applied Mechanics**, v. 5, n. 1, p. 180–191, 2024. DOI: `10.3390/applmech5010012`.

SAVITZKY, Abraham; GOLAY, Marcel J. E. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. **Analytical Chemistry**, v. 36, n. 8, p. 1627–1639, 1964. DOI: `10.1021/ac60214a047`.

SHIGLEY, Joseph E.; MISCHKE, Charles R. **Shigley's Mechanical Engineering Design**. [*S. l.*]: McGraw-Hill Education, 2015.

SMITH, Steven W. **The Scientist and Engineer's Guide to Digital Signal Processing**. [*S. l.*]: California Technical Publishing, 1997.

STREAMLIT COMMUNITY. **Streamlit Documentation**. [*S. l.: s. n.*], 2026. urlhttps://docs.streamlit.io/. Accessed: 2026-02-19.

TLUSTY, Jiri. **Manufacturing Processes and Equipment**. [*S. l.*]: Prentice Hall, 2000.

WIKIPEDIA CONTRIBUTORS. **Delta Air Lines Flight 1288**. Available from: `https://en.wikipedia.org/wiki/Delta_Air_Lines_Flight_1288`. Acesso em: 28 jan. 2026.

YIN, Qingan. **Recent progress of machinability and surface integrity for mechanical machining Inconel 718: a review**. Available from: `https://www.researchgate.net/publication/342608798_Recent_progress_of_machinability_and_surface_integrity_for_mechanical_machiningInconel_718_a_review`. Acesso em: 28 jan. 2026.

APPENDIX A – CORNER FINDING FUNCTION

```
def angle_between(v1, v2):
    cos_theta = np.dot(v1, v2) / (np.linalg.norm(v1) * np.
        linalg.norm(v2) + 1e-8)
    return np.arccos(np.clip(cos_theta, -1.0, 1.0))  # In
        radians

def find_corners(x, y, z, t):
    t_numeric = (t - t.iloc[0]).dt.total_seconds().values
    vx = np.gradient(x, t_numeric)
    vy = np.gradient(y, t_numeric)
    vz = np.gradient(z, t_numeric)
    # Velocity vector at each time step
    v = np.stack([vx, vy, vz], axis=1)
    angles = np.array([angle_between(v[i], v[i+1]) for i in
         range(len(v)-1)])
    angles_deg = np.degrees(angles)
    corner_indices = np.where(angles_deg > 50)[0]
    angle_times = t_numeric[:-1]

    return corner_indices, angles_deg
```

APPENDIX B – SAVITSKY-GOLAY FILTER IMPLEMENTATION

```
def plot_load_and_forces(df):
    with st.spinner("Breaking␣Tools..."):
    #give option to show for all operations
        if type(df) is type(None):
            st.warning("Some␣operations␣might␣not␣have␣passed␣
               through␣engagement␣properly.")
            return
        for index in df.index:
            with st.expander(f"Operation:␣{df['operation'][
               index]}"):

                median = np.median(spindle_load)
                std = np.std(spindle_load)
                max_val = np.max(spindle_load)

                # Define thresholds adaptively
                min_height = median + 0.5 * std
                min_prominence = 0.75 * (max_val - median)
                min_distance = int(0.1 * len(spindle_load))


                # robust baseline + adaptive peak detection
                try:
                    # estimate typical sample interval in
                       seconds
                    if len(timestamp) > 1:
                        dt_ms = np.median(np.diff(timestamp).
                           astype("timedelta64[ms]").astype(
                           float))
                        dt_s = max(dt_ms / 1000.0, 1e-6)
                    else:
                        dt_s = 0.01

                    total_dur_s = (timestamp[-1] - timestamp
                       [0]).astype("timedelta64[ms]").astype(
                       float) / 1000.0 if len(timestamp) > 1
```

```
    else len(spindle_load) * dt_s

# adaptive minimum distance between peaks (
    e.g., 0.1 s or 1% of total)
min_distance_sec = max(0.05, 0.001 *
    total_dur_s)
min_distance = max(1, int(min_distance_sec
    / dt_s))

# baseline / lowpass: prefer savgol (
    preserves edges). Fallback to moving
    average.
baseline = None
try:
    # choose window as ~1–2% of signal
        length (smaller window), odd and >=5
    win = max(5, int(len(spindle_load) *
        0.01))   # decreased from 0.03 to
        0.01
    if win % 2 == 0:
        win += 1
    if win >= len(spindle_load):
        win = len(spindle_load) - 1
        if win % 2 == 0:
            win -= 1
    if win < 5:
        win = 5
    # increase polynomial order but ensure
        it's < window size
    polyorder = 5
    if polyorder >= win:
        polyorder = max(3, win - 2)
    baseline = savgol_filter(spindle_load,
        window_length=win, polyorder=
        polyorder, mode="interp")
except Exception:
    # fallback moving average
    k = max(3, int(len(spindle_load) *
        0.02))
```

```
        kernel = np.ones(k) / k
        baseline = np.convolve(spindle_load,
            kernel, mode="same")

    residual = np.array(spindle_load) -
        baseline

    # robust threshold on residual using MAD
    mad = median_abs_deviation(residual) if len
        (residual) > 0 else 0.0
    med_res = np.median(residual) if len(
        residual) > 0 else 0.0

    # secondary threshold using percentile of
        raw signal to avoid constant-high flags
    high_pct = np.percentile(spindle_load, 90)
        if len(spindle_load) > 0 else 0.0
    global_scale = max( (high_pct - np.median(
        spindle_load)) * 0.3, 0.0)


    # final thresholds
    height_thresh = max(med_res + 5 * mad,
        global_scale * 1.0, 0.0)  # increased
        MAD multiplier from 3->5 and global
        scale weight
    prominence_thresh = max(3 * mad, 0.03 * (np
        .max(spindle_load) - np.min(spindle_load
        )))  # increased prominence requirement

    # find candidate peaks on residual (peaks
        above baseline)
    peaks_indices, props = find_peaks(
        residual,
        height=height_thresh,
        prominence=prominence_thresh,
        distance=min_distance
    )
    # compute widths (in samples) at half-
```

```
    prominence
if len(peaks_indices) > 0:
    widths_res = peak_widths(residual,
        peaks_indices, rel_height=0.5)[0]
else:
    widths_res = np.array([], dtype=float)
# minimum width: e.g., 0.01 s or at least 1
    sample
min_width_sec = 0.01
min_width_samples = max(1, int(
    min_width_sec / max(dt_s, 1e-6))) - 0.03
    #numero colocado pra passar um ponto no
    reamer
# margin required above baseline / filtered
    signal
filter_margin = max(3 * mad, 0.02 * (np.max
    (spindle_load) - np.min(spindle_load)))
validated = []
for i, idx in enumerate(peaks_indices):
    raw_val = float(np.array(spindle_load)[
        idx])
    base_val = float(baseline[idx]) if
        baseline is not None else 0.0
    # get prominence if available
    prom = float(props["prominences"][i])
        if "prominences" in props else (
        raw_val - base_val)
    width = float(widths_res[i]) if i < len
        (widths_res) else 0.0
    height = float(props["peak_heights"][i
        ])
    # compare also with the provided
        spindle_load_filtered if available
        and aligned
    # require that the raw peak is NOT
        larger than the filtered signal (+
        margin).
    # If raw_val > filtered + margin we
        treat it as a false/artefact and
```

```
            reject. NOT USED!!
        filtered_ok = True
        # try:
        #     if spindle_load_filtered is not
            None and len(spindle_load_filtered)
            == len(spindle_load):
        #         filtered_ok = raw_val <= (
            float(spindle_load_filtered[idx]) +
            filter_margin)
        # except Exception:
        #     filtered_ok = True
        # validation criteria: prominence,
            width, and above baseline/filtered
        if (
            height > (height_thresh + 0.0001)
            and prom >= prominence_thresh
            and width >= min_width_samples
            and (raw_val - base_val) >= max(3 *
                mad, 0.01 * abs(base_val))
            and filtered_ok
        ):
            validated.append(idx)
    peaks_indices = np.array(validated, dtype=
        int)
    peak_positions = np.array(timestamp)[
        peaks_indices] if len(peaks_indices) > 0
         else np.array([])
    peak_values = np.array(spindle_load)[
        peaks_indices] if len(peaks_indices) > 0
         else np.array([])
```