Máster en Ingeniería Industrial

TRABAJO FIN DE MÁSTER

Lime Dosing System Upgrade

Autor: Darío Muñoz Hernández

Director: Javier Peiro Balaguer

Madrid

Julio de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Lime Dosing System Upgrade

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/25 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Darío Muñoz Hernández Fecha: 11/ 07/ 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Javier Peiro Balaguer Fecha: 11/ 07/ 2025

Máster en Ingeniería Industrial

TRABAJO FIN DE MÁSTER

Lime Dosing System Upgrade

Autor: Darío Muñoz Hernández

Director: Javier Peiro Balaguer

Madrid

Julio de 2025

Acknowledgements

I would like to express my sincere gratitude to the following individuals for their support throughout the development of this Project:

- **James Carver (EC&I Maintenance Manager)**, for teaching me FbCAD and Sequence CAD, the native programming languages of the DCS, and for his guidance on system integration.

- **Blair Ward (Deputy Operations Manager)**, for helping me understand the underlying chemistry of the process, as well as the operational details and compliance framework.

- **Stephen Lormor (General Manager)**, for facilitating my integration into FCC's EfW plant in Lincoln and trusting in my expertise to carry out this Project.

- **Javier Peiro (EfW UK Head)**, for his warm welcome and continued encouragement to develop and deliver this initiative.

Their combined support, knowledge, and trust were instrumental in the successful completion of this work.

RESUMEN DEL PROYECTO

Este proyecto presenta la automatización y optimización del sistema de dosificación de cal en una **planta de valorización energética de residuos (EfW)** ubicada en **Lincoln** (Lincolnshire, Reino Unido) y operada por **FCC Environment**. Se implementó con éxito una nueva estrategia de control en el **Sistema de Control Descentralizado (DCS)** de la planta, lo que resultó en un **aumento potencial del 31% en la eficiencia de la dosificación**. Bajo condiciones operativas equivalentes, esta mejora representa un ahorro anual potencial de aproximadamente **£124,000**.

Palabras clave: Automatización, Optimización, Energía, Residuos, Cal.

1. Introducción

Los sistemas de dosificación de cal desempeñan un papel fundamental en las plantas EfW, ya que son responsables de mantener las **emisiones de SO_2 y HCl** dentro de los límites regulatorios [1].

Antes del desarrollo de este proyecto, el proceso de dosificación de cal en la planta de Lincoln de FCC Environment [2] estaba solo parcialmente automatizado, dependiendo en gran medida de la experiencia de los operadores y de ajustes manuales. Este enfoque introducía variabilidad e ineficiencias en el uso de cal.

Para abordar estos desafíos, se desarrolló e implementó una **estrategia de control totalmente automatizada**, centrada en un **controlador PID** específicamente adaptado al comportamiento dinámico del sistema. Este controlador incorpora mejoras diseñadas para garantizar el cumplimiento normativo constante mientras se optimiza el consumo de reactivos.

2. Definición del Proyecto

El objetivo de este proyecto fue **automatizar y optimizar el sistema de dosificación de cal** en la planta EfW de FCC Environment en Lincoln (Lincolnshire, Reino Unido). El proyecto incluyó el diseño, simulación e implementación de una solución de control personalizada dentro del Sistema de Control Descentralizado (DCS) de la planta, con el propósito de mejorar el rendimiento ambiental y reducir los costes operativos.

Los objetivos clave incluyeron:

- **Mantener las emisiones de SO_2 y HCl** dentro de los límites legales.
- **Reducir el consumo de cal** al menos en un 5%.

- **Minimizar la intervención del operador** mediante la automatización total del proceso.

3. Descripción del Sistema

El sistema mejorado de dosificación de cal se basa en una arquitectura de control PID (véase la Ilustración 1), implementada en el DCS utilizando programación en FbCAD y Sequence CAD. El controlador recibe datos en tiempo real sobre las concentraciones de SO_2 y HCl aguas arriba y el caudal de gases de combustión, calcula el requerimiento estequiométrico de cal y aplica un factor de corrección dinámico basado en desviaciones del proceso.

Las características clave del sistema incluyen:

- **Lógica de control Proporcional-Integral (PI)**, optimizada mediante simulación y pruebas.
- Mecanismo **anti-saturación** (**anti-windup**) para limitar el sobreimpulso de la integral durante desviaciones prolongadas.
- **Mejoras en la interfaz de operador** para la monitorización en tiempo real y la posibilidad de intervención manual.
- **Funcionalidad de reinicio de la integral** para asegurar la estabilidad tras paradas prolongadas.

El sistema ha demostrado un rendimiento sólido en condiciones operativas variables y puede adaptarse para su implementación en otras plantas EfW o ampliarse mediante métodos de control predictivo como la predicción basada en inteligencia artificial.

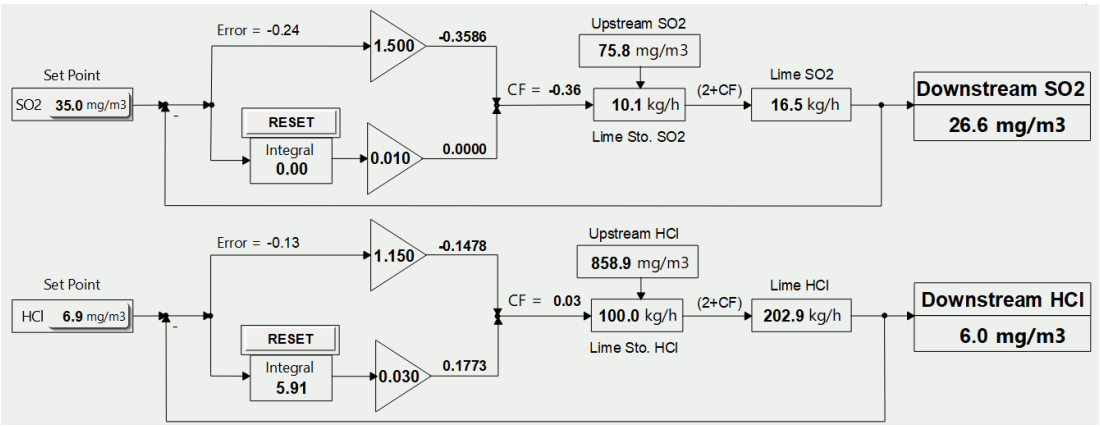


Ilustración 1 Diseño de los controladores PID de inyección de cal.

4. Resultados

La implementación de la nueva estrategia de control produjo mejoras significativas tanto en la eficiencia como en el cumplimiento normativo. Tras la puesta en marcha del controlador PID totalmente automatizado, la dosificación de cal se volvió más consistente, rápida y desvinculada de la intervención manual.

Los **principales resultados** incluyen:

- Una **reducción del consumo de cal** de **entre un 10% y un 31%** bajo condiciones operativas equivalentes.
- Un **ahorro anual estimado de entre £40,000 y £124,000**, derivado de un análisis económico integral basado en tres enfoques complementarios:
  1. Comparación de la **Relación Cal/Cal Estequiométrica (Lime2Sto)**.
  2. **Reducción directa del consumo de cal** en condiciones operativas equivalentes.
  3. Evaluación de la **Relación Cal/Residuos Incinerados**.
- **Emisiones de SO_2 y HCl mantenidas de forma constante por debajo de los límites regulatorios**, sin depender de factores de confianza aplicados por agencias medioambientales.
- Gracias a la **alta precisión del control**, se ajustó la configuración del controlador para regular en base a **emisiones corregidas** (**factores de confianza aplicados en tiempo real**), lo que permitió un ahorro adicional de cal superior al inicialmente previsto.
- **Mayor estabilidad del control**, con menos sobreimpulsos y oscilaciones mínimas en la dosificación de cal.
- **No se reportaron incidentes operativos** durante la puesta en marcha ni en la fase inicial de operación, según lo verificado en un **estudio HAZOP** realizado antes de la implementación.

Desde un punto de vista económico, el coste total del proyecto fue de **£3,377.50**. Asumiendo un escenario conservador de **£40,000/año** en ahorros y una **vida útil operativa de 5 años**, la inversión muestra un retorno extraordinario:

- **ROI**: 61
- **Tasa Interna de Retorno (TIR)**: 1,186%
- **Periodo de amortización**: ~1 mes

Estos resultados confirman la efectividad y robustez del nuevo controlador, validando su capacidad para permitir estrategias de optimización más agresivas sin comprometer el cumplimiento ambiental. La solución ofrece un gran potencial de replicabilidad en otras plantas EfW.

5. Conclusiones

Este proyecto cumplió con éxito todos los objetivos iniciales:

- Automatización del sistema de dosificación de cal.
- Reducción del consumo de cal más allá del objetivo del 5%.
- Cumplimiento continuo de la normativa sobre emisiones.

El diseño del controlador demostró ser robusto, eficiente y fácilmente interpretable por el personal de operación.

Aunque algunos desafíos, como los retardos del sistema y las no linealidades, no pudieron resolverse completamente, estos no comprometieron el rendimiento del sistema.

De cara al futuro, se proponen dos desarrollos adicionales:

- Extensión de la solución a otras plantas EfW, aprovechando su diseño modular y escalable.
- Integración de modelos predictivos basados en IA para anticipar las emisiones y compensar los retardos del proceso, mejorando aún más la precisión y eficiencia del control.

En conclusión, el proyecto representa una mejora técnica sólida y valiosa desde el punto de vista económico para el sistema de tratamiento de gases de combustión de la planta, con un alto potencial de aplicación en toda la industria.

6. Referencias

[1] «Energy from waste: a guide to the debate», GOV.UK. Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.gov.uk/government/publications/energy-from-waste-a-guide-to-the-debate

[2] «Homepage - FCC Environment». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.fccenvironment.co.uk/

LIME DOSING SYSTEM UPGRADE

Author: Muñoz Hernández, Darío**.**
Director: Peiro Balaguer, Javier.
Collaborating Entity: FCC Environment

PROJECT SUMMARY

This project presents the automation and optimisation of the **lime dosing system** at an **Energy-from-Waste (EfW)** facility located in **Lincoln** (Lincolnshire, UK) and operated by **FCC Environment**. A new control strategy was successfully implemented within the plant's Decentralised Control System (DCS), resulting in a **31% potential increase in dosing efficiency**. Under equivalent operating conditions, this improvement represents a potential **annual cost saving of approximately £124,000**.

Key Words: Automation, Optimization, Energy, Waste, Lime.

1. Introduction

Lime dosing systems play a critical role in EfW facilities as they are responsible for maintaining SO_2 and HCl emissions within regulatory limits [1].

Prior to the development of this project, the lime dosing process at FCC Environment´s [2] Lincoln plant was only partially automated, relying heavily on operator expertise and manual adjustments. This approach introduced variability and inefficiencies in lime usage. To address these challenges, a **fully automated control strategy** was developed and implemented, centred on a **PID controller** specifically tailored to the system's dynamic behaviour. This controller incorporates enhancements designed to ensure consistent regulatory compliance while optimising reagent consumption.

2. Project Definition

The objective of this project was to **automate and optimise the lime dosing system** at the FCC Environment EfW facility in Lincoln (Lincolnshire, UK). The project involved the design, simulation, and deployment of a custom control solution within the plant's Decentralised Control System (DCS), aiming to improve environmental performance and reduce operational costs.

Key goals included:

- **Maintaining SO_2 and HCl emissions** within legal limits.
- **Reducing lime consumption** by at least 5%.
- **Minimising operator intervention** through full process automation.

3. System Description

The upgraded lime dosing system is based on a **PID control architecture** (see **Illustration 1**), implemented in the DCS using **FbCAD** and **Sequence CAD** programming. The

controller receives real-time data on upstream SO_2 and HCl concentrations and flue gas flow, calculates the stoichiometric lime requirement, and applies a dynamic correction factor based on process deviations.

Key system features include:

- **Proportional-Integral (PI) control logic**, optimised through simulation and testing.
- **Anti-windup mechanism** to limit integral overshoot during prolonged deviations.
- **Operator interface enhancements** for real-time monitoring and manual override capability.
- **Integral reset functionality** to ensure stability after prolonged shutdowns.

The system has demonstrated strong performance across variable operating conditions and can be adapted for deployment in other EfW plants or extended through predictive control methods such as AI-based forecasting.

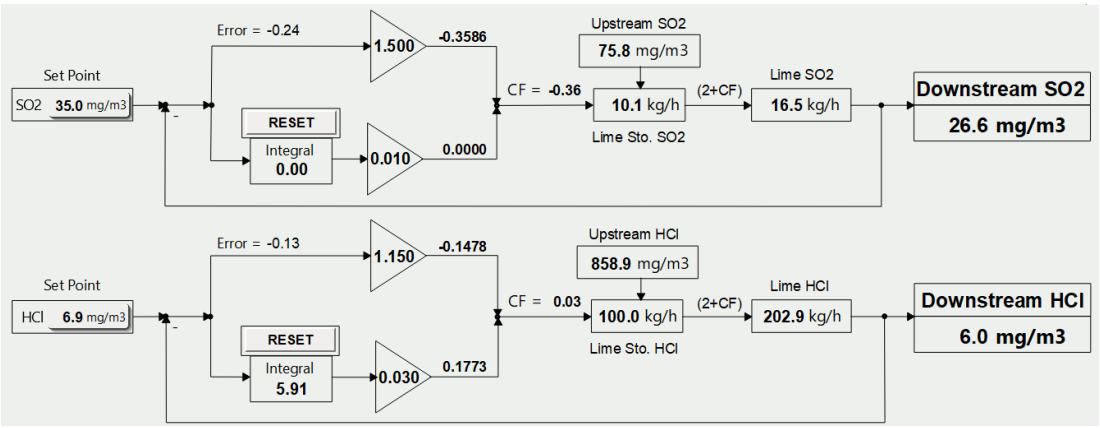


Illustration 1 Lime Dosing PID Control Layout.

4. Results

The implementation of the new control strategy yielded significant improvements in both efficiency and regulatory compliance. Following the deployment of the fully automated PID-based controller, lime dosing became more consistent, responsive, and decoupled from manual intervention.

Key results include:

- A reduction in lime consumption between **10% and 31%** under equivalent operating conditions.
- An **estimated annual cost saving between £40,000 and £124,000**, derived from a comprehensive economic analysis based on three complementary approaches:
    1. **Lime-to-Stoichiometric Lime Ratio (Lime2Sto)** comparison.
    2. **Direct lime consumption reduction** under equivalent operating conditions.

3. **Lime-to-Waste Ratio** assessment.

- **SO_2 and HCl emissions consistently maintained below regulatory limits**, without relying on environmental agency confidence factors.
- Due to the **high control accuracy**, the controller configuration was adjusted to regulate based on **corrected emissions (confidence factors applied in real time)**, enabling further lime savings beyond initial expectations.
- **Improved control stability**, with reduced overshooting and minimal oscillations in lime dosing.
- **No operational incidents** were reported during commissioning or early operation, as verified through a **HAZOP study** conducted prior to deployment.

From a financial standpoint, the total project cost was **£3,377.50**. Assuming a conservative scenario of **£40,000/year in cost savings** and a **5-year operational lifespan**, the investment demonstrates outstanding returns:

- **ROI**: 61.
- **Internal Rate of Return (IRR)**: 1,186%.
- **Payback Period**: ~1 month.

These results confirm the new controller's effectiveness and robustness, validating its capability to support more aggressive optimisation strategies while maintaining compliance with environmental standards. The solution offers strong potential for replication across other EfW plants.

5. Conclusions

This project successfully met all initial objectives:

1. **Automation** of the lime dosing system.
2. **Reduction of lime consumption** beyond the 5% target.
3. **Continuous compliance** with emission regulations.

The controller's design proved robust, efficient, and easily interpretable by operational staff. Although some challenges, such as system delays and non-linearities, could not be fully resolved, they did not compromise the performance of the system.

Looking forward, two future developments are proposed:

- **Extending the solution to other EfW plants**, leveraging its modular and scalable design.
- **Integrating AI-based prediction models** to anticipate emissions and compensate for process delays, further enhancing control accuracy and efficiency.

In conclusion, the project stands as a **technically sound and economically valuable upgrade** to the plant's flue gas treatment system, with high potential for wider application across the industry.

6. References

[1] «Energy from waste: a guide to the debate», GOV.UK. Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.gov.uk/government/publications/energy-from-waste-a-guide-to-the-debate

[2] «Homepage - FCC Environment». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.fccenvironment.co.uk/

Memory Index

Figure Index

Table Index

Section 1. INTRODUCTION

This project focuses on the **automation and optimization of a lime dosing system** of an **Energy from Waste** (**EfW**) plant in **Lincoln** (Lincolnshire, UK) operated by **FCC Environment**. This system plays a critical role in ensuring regulatory compliance by controlling **SO_2** and **HCl** within permitted limits.

Motivation

Prior to the development of this project, the lime dosing system was not completely automatic. It partially relied on the operators' understanding, ability, and analysis to dose an adequate amount of lime. This partial reliance on operators introduced inefficiencies that could be mitigated through full automation.

This presented an excellent opportunity to implement a more sophisticated solution which could potentially increase the efficiency of the lime use, increase the operators' productivity by reducing their tasks, and reduce the company's risk of breaching and getting economically sanctioned (as well as reputationally damaged).

Section 2. DESCRIPTION OF TECHNOLOGIES

The development of this Project relied on the implementation of the following technologies:

- **Visual Studio Code (VS Code)**: A lightweight, open-source code editor with support for debugging, version control, and extensions for various programming languages and tools.
- **VALMET DNA Explorer**: VALMET DNA Explorer is an engineering and monitoring interface for the Valmet DNA DCS [3], used to configure control strategies and analyse system performance in industrial environments. It allows DCS programming using proprietary tools: **FbCAD** and **Sequence CAD**.
- **DNA Operate Client**: A user interface of the Valmet DNA DCS for process operators to monitor, control, and interact with plant operations in real time, featuring alarm handling, trend views, and process graphics.
- **Microsoft Office Package**: Microsoft Excel was used for the handling of data, and Microsoft Word was used for the Operations Manual drafting.

Section 3. STATE OF THE ART

Prior to the development of this project, the lime dosing controller was not fully automatic. Its functioning was the following:

1. **Upstream measurements.**

When the flue gas from combustion gets to the lab loop, where the lime is dosed to react with the pollutants, sensors measure its SO_2 and HCl concentration in mg/Nm3. Additionally, flue gas flow (Nm3/h) is also measured to convert SO_2 and HCl concentration into mass flow (kg/h).

2. **Lime dosing stoichiometric calculation.**

Once the upstream measurements have been taken, the system calculates the theoretical lime needed (kg/h) to neutralize the pollutants down to specific set points given by the Environmental Agency.

3. **Correction factor application.**

Because the efficiency of the reaction is lower than 100%, correction factors are applied. There are a total of 3 correction factors:

- **30-Minute Set Point Deviation (K30)**: Measures the maximum deviation of the 30-minute average from the SO_2 and HCl set points.
- **24-Hour Set Point Deviation (K24)**: Measures the maximum deviation of the 24-hour average from the SO_2 and HCl set points.
- **Operator Factor ($K_{0\text{-}2}$)**: Consists of two **manual adjustment coefficients** (K_0 and K_2), introduced by the operator based on real-time trend analysis of downstream emissions. In practice, **K_0 is rarely modified**, making the system functionally equivalent to a **single-input control loop**.

The **final correction factor** used to modify lime injection is determined as the product of the K2 and the sum of K0, and the K24 and K30 deviation terms (see **Equation 1**):

$$Correction\ Factor = K_2 \cdot (K_0 + K_{30} + K_{24})$$

Equation 1 Legacy Correction Factor.

4. Downstream measurements and feedback.

At the end of the flue gas cycle, downstream emissions are recorded and fed back into the controller by recalculating the 30 minutes and 24 hours averages.
It should be noted that the downstream values which are fed back and reported to the Environmental Agency are applied other correction factors, given by this entity. The purpose for this is to standardize the emissions to a specific set of conditions (dry air and 11% O2 level) so that the mentioned entity can evaluate every EfW plant under the same conditions. Moreover, even though they do not take place in this controller, there are **confidence factors** being used in the Environmental Agency's evaluation that reduce the final **SO_2** and **HCl** emissions down to **80%** and **60%** respectively from their reporting.

It is also worth noting that emissions are subject to two separate reporting periods: a **30-minute report** and a **24-hour report**. The **24-hour report** enforces significantly more stringent setpoints – **8 mg/Nm³ for HCl** and **40 mg/Nm³ for SO_2**, after applying confidence factors. In contrast, the **30-minute report** allows for considerably higher limits, set at **60 mg/Nm³ for HCl** and **200 mg/Nm³ for SO_2**, providing a broader compliance margin over shorter timeframes.

A visual representation of the process has been included in **APPENDIX I** – Process Scheme.

Other Plants

Although similar in purpose, each EfW facility has its own lime dosing strategy, tailored to its specific layout, infrastructure, and technological constraints. A representative example is FCC Environment's EfW plant in Eastcroft (UK), the company's oldest plant in the country, which illustrates how technical limitations can hinder lime optimisation and challenge the replicability of this Project.

Several differences distinguish Eastcroft from Lincoln's facility:

- **DCS Software:** Eastcroft operates with a much older generation of DCS software. Its programming environment is significantly more restrictive, making loop visualisation difficult and requiring a rigid logic structure. Any error in the loop sequence could potentially crash the entire system. As a result, replicating this Project at Eastcroft would necessitate a meticulous testing procedure, including a formal HAZOP study.
- **Lack of Communication Between DCS and Analyser:** The current DCS architecture has no available input ports for additional communication. To enable data exchange, the system would require an upgrade – such as adding multiplexers, implementing serial communication protocols, or incorporating remote I/O modules networked with the DCS.
- **Lime Screw Limitations:** Unlike Lincoln's variable-speed screws (which can modulate lime dosing from 9% to 100%), Eastcroft's screws operate on a binary signal – either minimum or maximum dosing. To introduce modulation, a potential solution could be to implement a 5-minute pulse-width modulation (PWM) logic, allowing averaged dosing over the 30-minute reporting period.
- **Unavailable Feedback:** Due to the lack of communication with the analyser, the control logic currently applies maximum lime dosing when upstream pollutant levels exceed the setpoint. If downstream data were made available, a PID controller could be implemented to track the 30-minute moving average, enabling a dynamic and more efficient control similar to Lincoln's.

Scalability Considerations

The analysis of Eastcroft's facility demonstrates that while the control strategy developed in this Project is technically transferable, its successful replication across other EfW plants depends largely on each site's technological readiness. Key barriers such as outdated DCS platforms, limited I/O capacity, and mechanical dosing constraints must be addressed to ensure seamless integration.

Nevertheless, these challenges are not insurmountable. With targeted investments in communication infrastructure and control system upgrades, the PID-based controller and associated automation logic could be adapted and deployed effectively. The significant efficiency gains and cost savings achieved at Lincoln reinforce the value of pursuing such adaptations, particularly in facilities seeking to modernise their flue gas treatment systems and reduce operational expenditure.

In summary, the Project sets a solid foundation for a scalable control solution – provided that implementation is tailored to each plant's specific context and accompanied by adequate technical risk mitigation.

3.1 PROJECT DEFINITION AND JUSTIFICATION

Lime dosing systems in Energy-from-Waste (EfW) facilities often rely on manual intervention or basic fixed-parameter control strategies. These conventional approaches struggle to adapt to the dynamic and nonlinear behaviour of flue gas composition, which varies significantly due to changes in flow rate, pollutant concentrations (SO_2 and HCl), and other operational conditions. As a result, they frequently lead to inefficiencies – such as lime overdosing – and may compromise compliance with increasingly strict environmental regulations.

This project set out to develop and implement an advanced control solution tailored to these challenges: a **PID controller with gain scheduling**, capable of dynamically adjusting its response based on real-time variations in process variables. The gain scheduling strategy was originally selected to ensure optimal controller performance across different operating conditions, particularly in response to fluctuations in gas flow and pollutant load. At the design stage, the system's behaviour was assumed to be sufficiently nonlinear to justify this approach.

However, empirical testing during implementation revealed greater system stability across operating points than initially expected. As a result, the gain scheduling logic – though part of the initial architecture – was ultimately **simplified to a fixed-gain PID configuration**

without compromising performance. This adaptive shift in strategy demonstrates the flexibility and robustness of the control solution.

Despite not fully resolving two key design challenges:

- The **nonlinear behaviour** caused by variability in flue gas flow, temperature and pollutant concentrations, and
- The **process delays** introduced by transport lags in both actuator response and measurement acquisition –

the controller proved to be **effective and reliable**. These unresolved factors had **minimal impact on system performance** and did not prevent the achievement of the project's main goals:

- **Maintaining emissions below regulatory limits**.
- **Fully automating the lime dosing process**.
- **Reducing lime consumption by more than the targeted 5%**.

In addition, this solution provides a modular, easily integrable control structure that is adaptable to the specific needs of EfW facilities. It stands as a technically innovative and commercially attractive approach for DCS vendors, process engineers, and plant operators seeking to modernize and optimise their flue gas treatment systems.

3.2 OBJECTIVES

This Project intends to achieve the following objectives:

- **Keep SO_2 and HCl emissions under the established limits** to comply with regulations.
- **Full automation** of the lime dosing control.
- **Reduce lime consumption by** at least **5%**.

3.3 METHODOLOGY

The methodology followed to achieve this Project's goals involved a multi-step strategy encompassing the following stages:

System Model

To characterise the system's behaviour, an efficiency study was conducted using historical data extracted from the DCS. The process was then recreated in a Python script to simulate the neutralisation of acid gases under varying operating conditions.

Control Design

Based on the system model, a control strategy was developed and tuned through iterative simulations in Python. This phase enabled the validation of different tuning parameters and the evaluation of controller performance under realistic process scenarios.

Implementation

The final stage involved deploying the controller into the DCS environment by translating the Python-based algorithm into the system's native programming languages: **FbCAD** and **Sequence CAD**. The implementation follows standard control engineering practices for modular and safe automation system design. [4]

Additionally, a new user interface was developed to display real-time controller performance and to incorporate key control metrics.

To ensure operational safety and clarity, two critical activities were also conducted during this phase:

- A **Hazard and Operability Study (HAZOP)** to identify and mitigate potential risks associated with the new control system.
- The preparation of a comprehensive **Operations Manual** to support proper system use and long-term maintainability.

3.4 ECONOMIC PLANNING AND ESTIMATION

This section outlines the economic planning and resource estimation carried out throughout the project's development. It includes a Gantt chart detailing the scheduling of key activities and milestones (**Figure 1**), as well as a summary table of the methodology adopted which is presented in **Table 3** in **APPENDIX II** – Project Planification Summary Table.

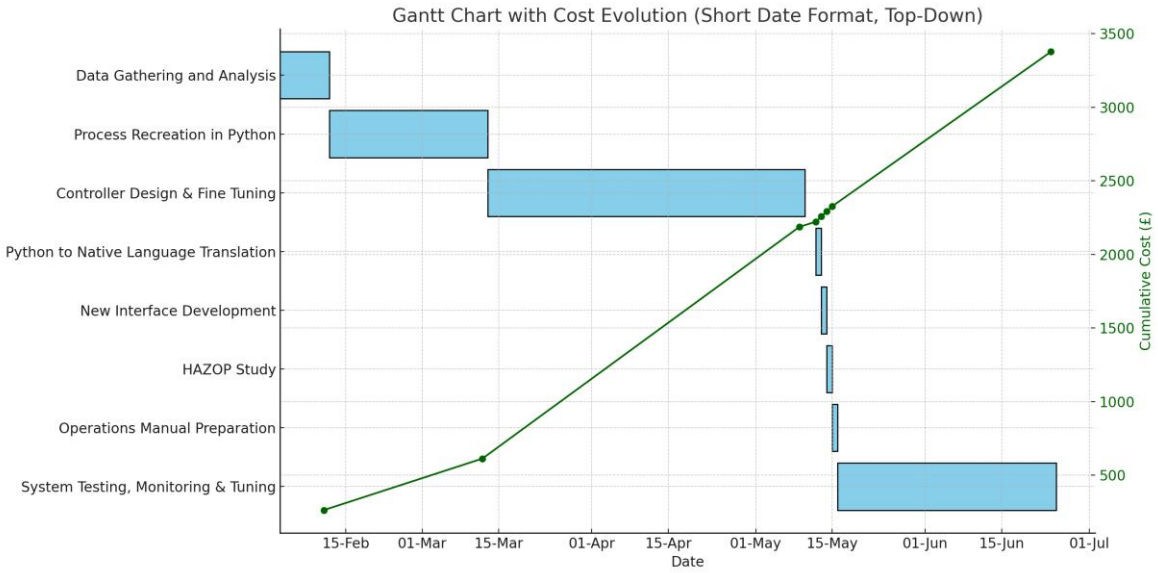


Figure 1 Project Timeline with Main Phases and Milestones.

Section 4. DEVELOPED SYSTEM

As previously described in **Section 3.3**, the Project was structured into 3 main phases:

- **System Model.**
- **Control Design.**
- **Implementation.**

This section offers a detailed overview of each phase, highlighting the methodology applied, techniques employed, and the key outcomes achieved at each stage of development.

4.1 SYSTEM MODEL

To characterize the system's behaviour, a global reaction efficiency analysis was first conducted, followed by an individual assessment of SO_2 and HCl neutralization efficiencies.

4.1.1 GLOBAL REACTION EFFICIENCY ANALYSIS

The global efficiency of lime dosing in reducing SO_2 and HCl concentrations was calculated using a dataset spanning one full year, with a sample time of 30 minutes. The analysis was based on historical process data extracted from the DCS, which included the following key variables:

- **SO2 & HCl Downstream Concentration [mg/Nm3].**
- **Flue Gas Flow [Nm3/h].**
- **Lime dose [kg/h].**

The objective was to evaluate how effectively the injected lime reduced the concentrations of SO_2 and HCl in the flue gas stream. This was done by computing the theoretical lime demand based on stoichiometry and comparing it to the actual lime consumption.

1. **Pollutant unit conversion (Equation 2)**: To determine the mass flow rate of each pollutant [kg/h], their concentrations [mg/Nm3] were multiplied by the flue gas volumetric flow rate [Nm3/h], and converted from milligrams to kilograms.

$$Pollutant_{[\frac{kg}{h}]} = Pollutant_{[\frac{mg}{h}]} \cdot \frac{Flue\ Gas\ Flow_{[\frac{m^3}{h}]}}{10^6_{[\frac{mg}{kg}]}}$$

Equation 2 Pollutant unit conversion from mg/Nm3 to kg/h.

2. **Theoretical lime needed (Equation 3)**: Based on the chemical reactions, 1 mol of $Ca(OH)_2$ neutralizes 1 mol of SO_2 and 2 mol of HCl [5]. Using the molar masses of each compound, the theoretical lime required to neutralize the removed fraction of SO_2 and HCl is calculated as:

$$Stoichiometric\ Lime\ Needed_{[\frac{kg}{h}]} =$$

$$= (SO2_{US} - SO2_{DS}) * \frac{MM_{Lime}}{MM_{SO2}} + (HCl_{US} - HCl_{DS}) * \frac{MM_{Lime}}{2 \cdot MM_{HCl}}$$

Equation 3 Theoretical lime needed based on stoichiometric equation.

3. **Global reaction's efficiency (Equation 4)**: The global reaction efficiency is defined as the ratio between the stoichiometric lime requirement and the actual lime dosed into the system.

$$Efficiency = \frac{Lime_{Stoich.}}{Lime_{dosed}}$$

Equation 4 Global reaction's efficiency.

The result of this study concluded that the **global reaction efficiency was 32%**, indicating that 32% of the total lime injected was effectively used for the neutralization of SO_2 and HCl.

4.1.2 INDIVIDUAL REACTION EFFICIENCY AND TRANSPORT DELAY ESTIMATIONS

Because it is not possible to know the proportions which the lime reacts with each pollutant, an estimation is needed. The estimations were done through 2 different procedures:

4.1.2.1 SO2 Efficiency and Transport Delay Estimation

To estimate the SO_2 neutralization efficiency and characterize the system's transport delay, the script ***Efficiency_Estimation.py*** was developed to execute simulations using plant data collected from the DCS. The datasets used for this analysis consist of daily records from different months, each with a consistent sampling interval of **10 seconds**, enabling a high temporal resolution for transient behaviour analysis.

The mentioned script has been presented in **APPENDIX III** – Efficiency_Estimation.py.

Methodology

1. **Data preprocessing**

The script loads and sanitizes **CSV files** with relevant process variables, including **upstream SO_2 concentration, lime injection rate, flue gas flow, moisture (H_2O), and oxygen (O_2 dry) levels**. Only valid numerical entries are retained for analysis.

2. **Unit Conversions and Corrections**

- **Upstream SO_2 concentrations [mg/Nm³]** are **converted** to **mass flow rates [kg/h]** using the **volumetric flow of flue gases [Nm³/h]**.
- **Stoichiometric neutralization** is approximated assuming a **fixed efficiency for SO_2** removal per kg of lime injected.
- The **downstream SO_2 mass flow** is estimated by **subtracting** the **neutralized** amount from the **upstream mass flow**, ensuring no negative concentrations.
- This value is then **normalized** back to **[mg/Nm³]** and **corrected** for humidity and oxygen concentration based on standard reference conditions (**dry air, 11% O_2**).

3. Delay Modelling

A step delay is applied to downstream SO_2 concentration values to **represent the gas transport delay between the injection and measurement points**. Since the data sampling interval is 10 seconds, **each step** corresponds to a **10 second delay**. Therefore, a delay of 6 steps effectively simulates a 60-second transport delay, which accounts for the time required for the flue gas to travel from the point of lime injection to the location where the downstream SO_2 concentration is measured.

4. Comparison with Measured Data

The estimated downstream SO_2 concentrations are compared against the actual measured values. The alignment between simulated and measured signals enables estimation of the neutralization efficiency and validates the assumed delay parameters.

5. Visualization

The results are plotted over a user-defined interval, typically showcasing the **estimated vs. real downstream SO_2 concentrations** along with the lime injection profile. This visual comparison facilitates performance evaluation and model calibration.

The estimation of SO_2 reaction efficiency and transport delay was conducted by visually comparing the model's output against real downstream SO_2 measurements across several representative days, each consisting of approximately 8600 samples (10-second intervals). For clarity and better visualization, **Figure 2** displays a **1000-step interval** (approximately 2.8 hours) from one selected day. The model uses a **fixed reaction efficiency of 4%** and a **transport delay of 6 steps (60 seconds)**. Within the displayed interval, the estimated SO_2 concentration closely follows the trend of the measured values, supporting the adequacy of the chosen efficiency and delay parameters. The corresponding lime dosing rate is also plotted to contextualize the system's response to injection events.

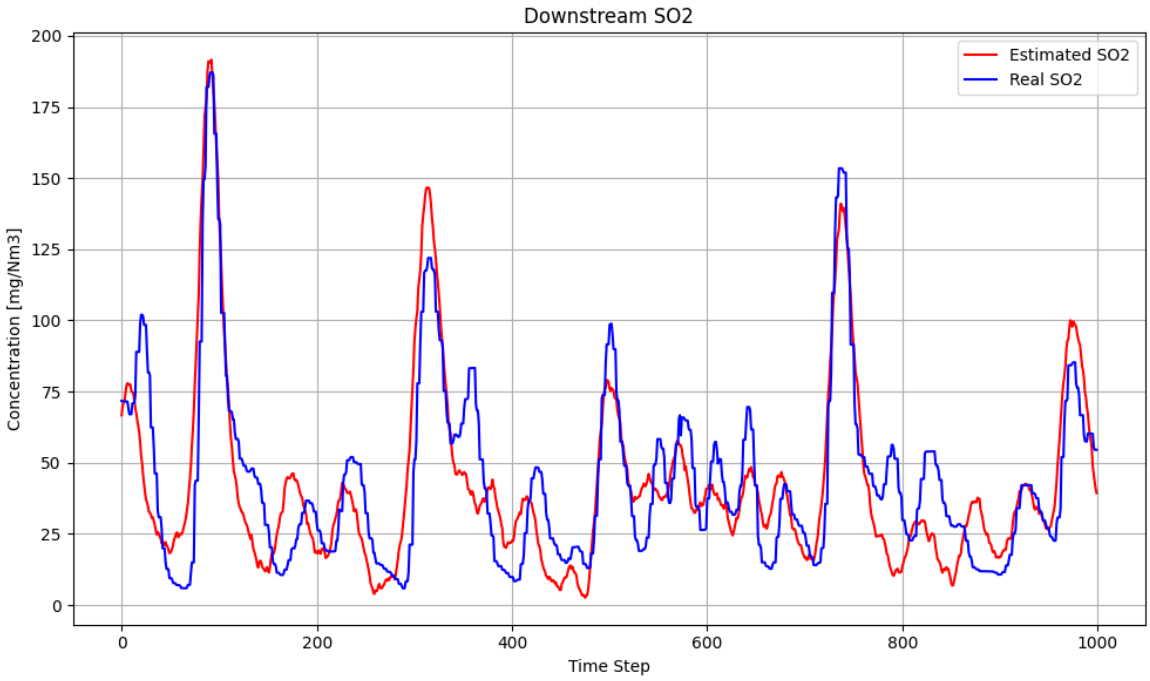


Figure 2 Comparison between simulated and real downstream SO_2 concentrations using a reaction efficiency of 4% and a transport delay of 6 steps (60 seconds).

4.1.2.2 HCl Efficiency Estimation

Given that the global lime usage efficiency over the dataset is 32%, and the estimated SO_2 efficiency is only 4%, we infer that SO_2 accounts for a minor fraction of total lime consumption. This implies that most of the effectively used lime was dedicated to neutralizing HCl. By subtracting the portion attributed to SO_2 from the global reacted lime, we can estimate the efficiency associated with HCl neutralization. **This approach assumes that the SO_2 and HCl neutralization reactions are independent and do not chemically interfere with each other** — a valid assumption given their distinct reaction pathways and kinetics. Therefore, the remaining efficiency (approximately 28%) can be attributed to HCl, from which an individual efficiency can be computed relative to its theoretical lime requirement.

4.1.3 FINAL MODELLING

Following the estimation of neutralization efficiencies and the characterization of transport delay in downstream emissions, the remaining components of the model were implemented to accurately replicate system behaviour. The modelling workflow is structured as follows:

Lime Dosing Calculation

The **required lime dosing rate [kg/h]** is determined based on the **stoichiometric requirements** for SO_2 and HCl neutralization. This calculation includes the application of a **correction factor**, which adjusts the theoretical dosage based on emission deviations from setpoints. The derivation and tuning of this correction factor are further detailed in the subsequent section.

Neutralised Pollutants Estimation

The **mass of SO_2 and HCl neutralized per unit time** is calculated using the lime dosing rate, the molar masses of the involved compounds, and the previously estimated neutralization efficiencies. This step models the chemical reaction behaviour of lime with each pollutant.

Downstream Emissions Calculation

The **downstream pollutant mass flows [kg/h]** are obtained by subtracting the neutralized pollutant amounts from their respective upstream mass flows. These values are subsequently **converted to concentration units [mg/Nm³]** by normalizing with the flue gas volumetric flow rate.

Standardisation of Emissions

The downstream concentrations are adjusted using **correction factors** that **normalize** the **gas conditions to 11% reference oxygen and dry basis**, in compliance with **environmental regulatory standards**. This ensures the resulting emission values are consistent with those reported to the Environmental Agency.

Transport Delay Simulation

A first-in, first-out (FIFO) buffer is implemented using a ***deque array*** **to simulate the transport delay of emissions from the injection point to the monitoring point**. This buffer ensures temporal alignment between the dosing action and its delayed effect on downstream measurements.

4.2 CONTROLLER DESIGN

The development of the lime dosing controller followed a structured approach that combined controlled simulation environments with real-world validation. To facilitate this, three dedicated scripts were developed – each serving a specific role in the design and evaluation process.

The first script, ***PID_Tunning_StepSim.py*** (see **APPENDIX IV** – PID_Tunning_StepSim.py) was used to **fine-tune the PID controller gains**. It simulated idealized conditions by applying step changes to upstream SO_2 and HCl concentrations, helping to identify appropriate gain scheduling strategies and to evaluate the effect of system delays. This environment allowed for controlled experimentation and systematic adjustment of the controller parameters.

The second script, ***Controller_Validation_RealData.py*** (see **APPENDIX V** – Controller_Validation_RealData.py) focused on **testing the tuned controller against actual operating conditions**. It loaded historical data from the plant's DCS covering 12 representative days (one per month) allowing comparisons between the simulated controller outputs and real plant behaviour in terms of emissions and lime consumption. In addition, the third script, ***Daily_ControllerVisualizer.py*** (see **APPENDIX VI** – Daily_ControllerVisualiser), was developed to enable users to **simulate and visualise** the system's performance over a **single day of their choice**. This tool facilitates focused analysis on specific operational scenarios, supporting further validation and operator insight. The following subsections describe each of these tools in more detail.

4.2.1 PID GAIN TUNING THROUGH STEP-BASED SIMULATION

To design an effective controller for the system, the script ***PID_Tunning_StepSim.py*** (see **APPENDIX IV** – PID_Tunning_StepSim.py) was developed to simulate generated step changes in the upstream emissions of SO_2 and HCl. These step values were selected based on recorded data from a year-long dataset, ensuring that the simulation covered a wide range of operating points. This variety was crucial for the design process, as it allowed the controller to be optimized for different conditions.

Initially, experimentation was performed using a standard PID controller. However, after observing that the PID controller's performance varied significantly across different operating points, a decision was made to transition to a **gain scheduling PID**. This approach dynamically adjusted the controller's parameters according to the current operating point. The operating points for gain scheduling were selected based on the experimental (simulated) results, ensuring that the controller adapted effectively to different system behaviours.

The controller logic is based on the **independent regulation of SO_2 and HCl emissions**. **Two separate PID controllers were implemented** – one for each pollutant – with tailored gain parameters to reflect their differing dynamics and neutralization requirements. The lime needed to neutralize SO_2 and HCl is calculated separately and then combined to determine the total dosing.

Furthermore, the **reading delay** in the system – primarily due to flue gas transport time and measurement lags – was accounted for and simulated in the controller model. To mitigate the effects of this delay, the **Smith Predictor** [6] technique was applied, which helped anticipate the delay and correct for its impact. The effects of delay misestimation were also explored, providing valuable insights into the system's robustness and sensitivity.

In parallel with the feedback control strategy, a **feedforward component** was integrated into the controller. This feedforward path was based on the measured upstream concentrations of SO_2 and HCl, the stoichiometry of the neutralization reactions, and the flue gas flow rate. It

allowed the controller to proactively compute the required amount of lime before the emissions reached the measurement point, thereby improving response time and reducing overshoot. The feedforward output was combined with the feedback control effort to compute the final lime dosing rate, ensuring both anticipatory and corrective control actions were taken.

Additionally, a key mechanism was introduced to avoid interference between the two controllers: **integral reset logic**. If one pollutant requires more lime (positive integral action) while the other is well below its setpoint (negative accumulation of errors), the integral of the second controller is reset to zero to prevent counterproductive interaction. For example, if SO_2 concentrations are above the setpoint and require additional lime, but HCl is significantly below its setpoint, the HCl controller may try to reduce dosing. If both integrals were allowed to accumulate, the negative integral from HCl could offset the positive dosing demand from SO_2, leading to underdosing and prolonged exceedance of the SO_2 emission limits. The reset condition avoids such conflict and ensures that lime dosing is never suppressed when one pollutant still requires neutralization.

The result of the gains' fine tuning can be observed in **Figure 3**.

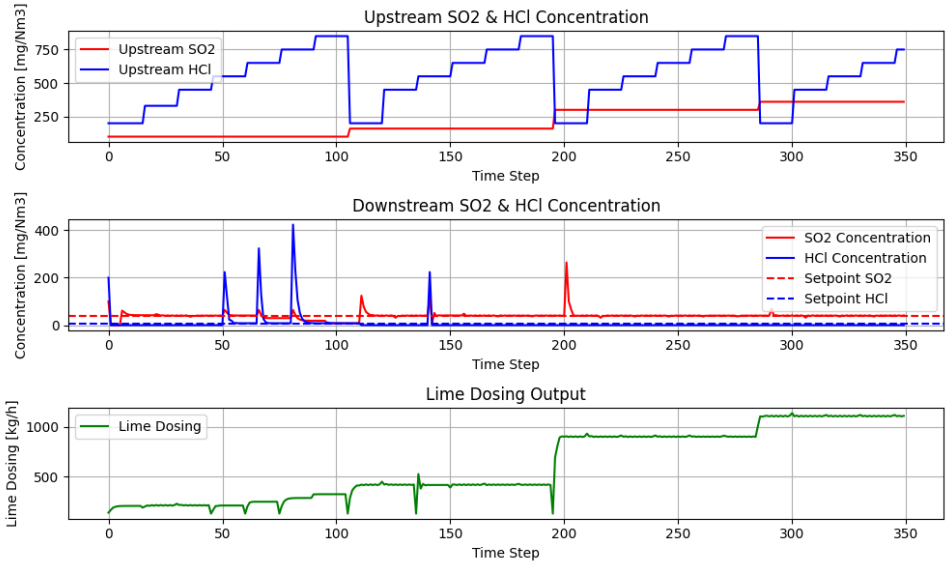


Figure 3 Final result after fine tuning the PID gains by observing step responses.

4.2.2 CONTROLLER VALIDATION WITH HISTORICAL PLANT DATA

To evaluate the controller's performance under realistic conditions, the script ***Controller_Validation_RealData.py*** (see **APPENDIX V** – Controller_Validation_RealData.py) was developed to simulate the system using historical data extracted from the DCS. This dataset consisted of 12 distinct days, one per month, selected to represent a broad range of plant operating conditions and seasonal variability throughout the year. The intention behind this selection was to ensure that the controller's robustness and effectiveness could be validated under diverse scenarios.

In the simulation, the real measured upstream SO_2 and HCl concentrations, as well as the flue gas flow and other relevant process variables, were used as inputs. The script computed what the lime dosing would have been if the new controller (including both feedforward and feedback components) had been in place. This simulated lime dosing was then compared directly to the actual lime dosing recorded by the DCS.

To assess the outcome of the simulated controller relative to real plant performance, the script produced several key comparisons:

- A **summary** of the **average SO_2 and HCl emissions** over each of the 12 days, both for the real system and the simulation.
- A **summary** of the **average lime consumption** under both conditions.
- **Visual plots** contrasting the real lime dosing curve with the simulated one, helping to illustrate dynamic differences in dosing behaviour.
- A **final plot** that presented the **real vs simulated emissions**, showing how close they were to the target setpoints.

In the last stage of analysis, environmental compliance confidence factors – issued by the environmental agency – were applied to both real and simulated emissions. These factors account for uncertainties in measurements and define the upper bounds for compliance. This allowed the script to determine how the simulated emissions would be interpreted under regulatory scrutiny and how they compared to the emissions setpoints. The visualization

offered insights into the controller's potential for maintaining compliance more efficiently or with reduced lime usage and helped quantify the trade-offs between emission control and reagent consumption.

After the final tuning, the script returned the following summary:

```
Mean SO2 emissions: 26.09 mg/Nm3 vs Actual: 35.98 mg/Nm3
Mean HCl emissions: 5.49 mg/Nm3 vs Actual: 7.93 mg/Nm3
Mean Lime Dosage: 267.03 kg/h vs Actual: 250.51 kg/h
```

These preliminary results suggest that the new controller achieves **lower emissions** but at the cost of **increased lime consumption**. The controller appears to overreact, especially during transients, resulting in overshoots that lead to higher reagent use.

Visual Analysis

Figure 4 simulates the **simulated behaviour** of the new controller on the first day of April. The plot highlights **significant overshooting** in the simulated lime dosing curve, consistent with the observed overconsumption.

Figure 5 compares the **24 hours average emissions** before (continuous lines) and after (discontinuous lines) applying the Environmental Agency's confidence factors. The original controller, while seemingly compliant, operates close to the regulatory thresholds and would exceed them without the benefit of these factors. In contrast, the new controller maintains **emissions well below the limit**, providing a **greater margin of compliance**.

These insights suggest that the new controller offers superior **robustness and reliability**, even though the initial summary implies higher average deviations from setpoints. This apparent contradiction arises from the trade-off between variables – when one pollutant is better controlled, the other may deviate more.

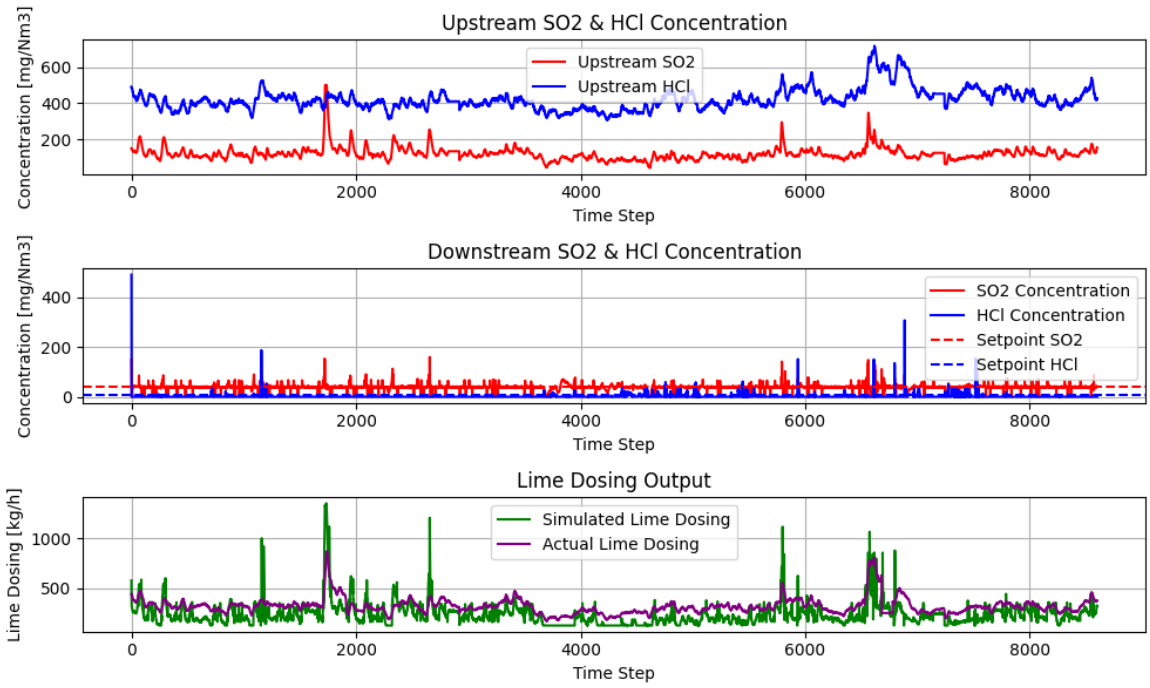


Figure 4 Visual plot for the simulated day in April.

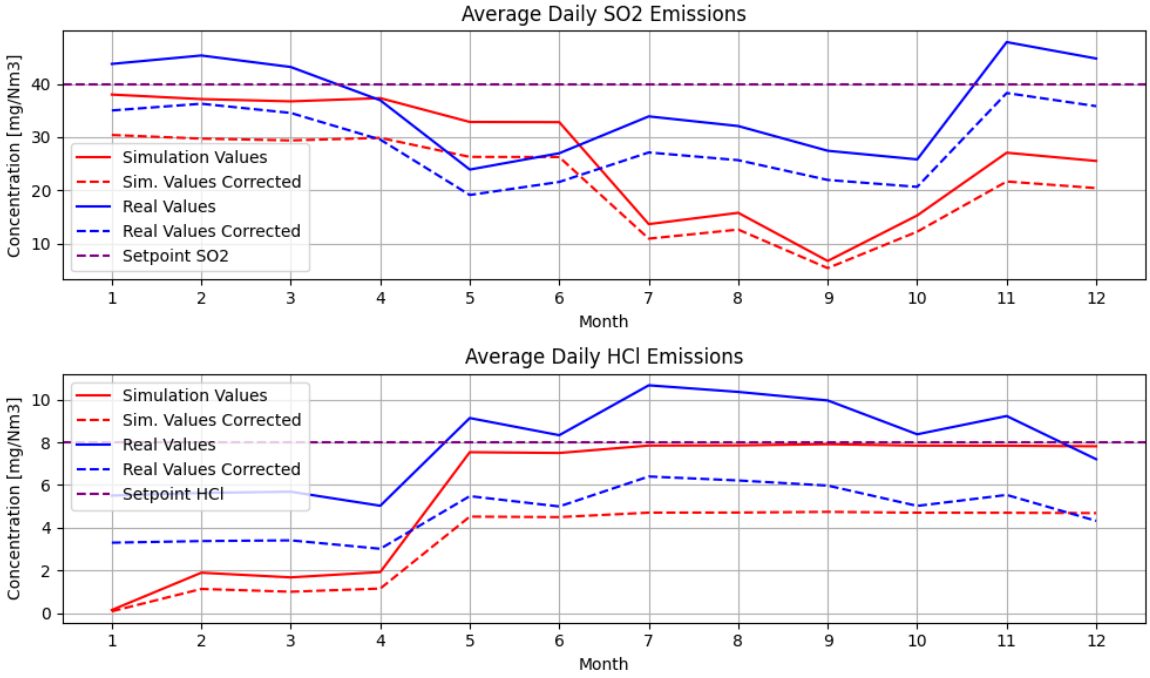


Figure 5 Visual Representation for Average Daily Emissions Outcome.

Adjusted Assessment Using Confidence Factor-Adjusted Setpoints

To better reflect the new controller's capabilities, the setpoints were recalculated by dividing the original values by the corresponding confidence factors. This yielded the following adjusted results:

```
Mean SO2 emissions: 31.96 mg/Nm3 vs Actual: 35.98 mg/Nm3
Mean HCl emissions: 10.33 mg/Nm3 vs Actual: 7.93 mg/Nm3
Mean Lime Dosage: 246.34 kg/h vs Actual: 250.51 kg/h
```

Under this adjusted framework, the new controller demonstrates the potential for **lower lime consumption**, while maintaining emissions within compliant margins. However, the results indicate that it does **not achieve the 5% lime reduction target**, although it does approach it more closely than the original summary suggested.

4.2.3 SOFTWARE ARCHITECTURAL DESCRIPTION

This section outlines the architectural description of the scripts described previously.

4.2.3.1 Efficiency_Estimation.py – Static Estimation of Neutralization Efficiency

This script was designed to estimate the chemical neutralization efficiency of SO_2 using lime, and to assess the delay present in the downstream measurement of emissions. To achieve this, an iterative testing process was followed, where multiple values were trialled for the SO_2 reaction efficiency and the downstream delay – the latter adjusted in 10-second increments. The HCl efficiency was previously estimated and used as a fixed value during the simulation.

The script operates on real plant datasets containing flue gas parameters, pollutant concentrations, and lime injection rates. It reproduces downstream SO_2 and HCl concentrations based solely on a mass balance model, assuming no active control intervention. The outcome is compared against real downstream measurements to evaluate how much of the emission reduction can be attributed to the chemical action of lime.

Code Structure and Functional Components

Although the script does not define custom classes, it is functionally organized into sequential blocks that each fulfil a specific role:

1. Constants and Imports

Standard Python libraries are imported, including:

- `numpy, matplotlib, pandas` for numerical computation and plotting,
- `glob, natsort` to load and sort files,
- `deque` from `collections` to implement delay buffers.

2. Dataset Loading and Cleaning

The script uses the following logic to import and prepare data:

```
file_list = glob.glob("limeDosingControl_*.csv")
file_list = natsorted(file_list)
selected_file = file_list[dataset_index]
data = pd.read_csv(selected_file)
```

- The user is prompted to select a dataset interactively.
- A subset of columns is selected, including:
- SO2 Upstream, HCl Upstream, AirFlow, Lime Injected, H2O, O2 dry, and corrected downstream emissions.
- Invalid entries (#REF!, NaN, etc.) are removed.
- All values are converted to NumPy arrays for efficient use in the simulation loop.

3. Main Simulation Loop

The core logic resides in a for loop that simulates the neutralization process over 8600 time-steps (approximately 24 hours):

```
for step in range(steps):
    ...
```

At each step:

- **Inputs**: Upstream SO_2 and HCl concentrations, airflow, and actual lime dosing are read from the dataset.
- **Conversion**: Emissions are converted from [mg/Nm³] to [kg/h] using the airflow.
- **Neutralization Calculations**:
  - SO_2 and HCl neutralized are estimated using:

    ```
    neutralized_SO2 = lime_dose * 64 / 74 * efficiency_so2
    neutralized_HCL = lime_dose * (36.45 * 2) / 74 * efficiency_hcl
    ```

    These stoichiometric formulas convert lime mass to pollutant removal assuming predefined efficiencies.
- **Downstream Estimation**:
  - The downstream emissions are calculated by subtracting the neutralized values from the upstream values.
  - Results are transformed back to [mg/Nm³] and corrected using standard factors for H_2O and O_2:

    ```
    cf_h2o = 100 / (100 - h2o_level)
    cf_o2 = (20.95 - 11) / (20.95 - o2_level)
    ```

4. Delay Implementation:

A delay buffer is implemented using deque to simulate sensor lag:

```
downstream_SO2_d = deque([0] * delay_reading, maxlen=delay_reading)
...
downstream_SO2_d.append(downstream_SO2)
downstream_so2_values.append(downstream_SO2_d[0])
```

This allows the comparison between **real delayed readings** and simulated outputs to be aligned in time.

5. Result Visualisation

The script produces comparative plots to visually assess the model's accuracy:

```
plt.plot(downstream_so2_values, label="Estimated SO2")
plt.plot(so2_values_ds, label="Real SO2")
plt.plot(lime_dosing, label="Lime dose")
```

Similar plots are generated for HCl, allowing analysts to visually inspect model errors and refine delay/efficiency estimates as needed.

4.2.3.2 PID_Tunning_StepSim.py – Controller Tuning through Step-Based Scenario Simulation

This script is intended for testing and tuning the PID-based lime dosing controller under synthetic step-like disturbances in SO_2 and HCl upstream concentrations. By isolating the system from real plant noise and variability, it provides a controlled environment for evaluating the **controller's responsiveness**, **stability**, and **robustness** to sudden changes in emission levels.

It is typically used before deploying the controller on real datasets to ensure that the gain scheduling logic and the PID behavior are well-calibrated across a wide operating range.

Overview of the Simulation Framework

Unlike the previous script (`Efficiency_Estimation.py`), which relies on real measured inputs, `PID_Tunning_StepSim.py` **generates its own emission profiles** using synthetic step changes. The simulation reproduces the system's response over a series of time steps using a **combined feedforward + PID control structure**, together with a simplified neutralization model.

Key Modules and Functional Elements

1. Constants and Configuration

```
MAX_SCREW = 1380 * 1
MIN_SCREW = 1380 * 0.09
```

These values define the physical limits of lime injection and are used to constrain the controller output using `np.clip`.

2. PID Controller Class (`PIDController`)

This custom class implements the full logic of a PID controller, including:

- Proportional, Integral, and Derivative terms.
- Gain updating (`update_gains`) for dynamic tuning.
- Integral reset logic to avoid windup or interference when switching gain sets.
- Optional integral clamping via `integral_limit` (though it is not enforced in all cases).

3. Step Input Generator (`simulate_steps`)

This function defines a sequence of emissions step changes over time. It updates the values of upstream SO_2 and HCl depending on the current simulation step:

```
if step >= 15:
    upstream_so2 = 100
    upstream_hcl = 330
...
```

The logic introduces sequential disturbances in a staircase pattern, testing the controller's adaptability to abrupt emission changes.

4. Gain Scheduling (`gain_schedule`)

Implements dynamic gain selection based on current upstream SO_2 and HCl concentrations. It returns distinct sets of (Kp, Ki, Kd) values depending on the concentration ranges:

```
Kp_SO2, Ki_SO2, Kd_SO2 = find_gains(upstream_SO2, so2_ranges)
```

This ensures that the controller behaves appropriately across a wide emission range, enhancing both **stability** in low loads and **aggressiveness** under high loads.

5. Feedforward and Lime Dosing Logic

The script uses a combination of:

- **Feedforward calculation**, based on stoichiometric requirements from deviations to the setpoints.
- **PID corrections**, computed from the downstream error (including a correction for transport delay).

- **Decoupled lime dosing**, where the SO_2 and HCl contributions are computed separately and summed.

```
lime_needed_SO2 = (QSO2 / 64) * 74 * (1 + cf_SO2 - C_HCL * cf_HCL)
lime_needed_HCL = (QHCL / (36.45 * 2)) * 74 * (1 + cf_HCL - C_SO2 * cf_SO2)
```

These expressions reflect the proportional relationship between pollutant loading and lime requirement.

6. Simulation Loop (`simulate_pid_controller`)

The main loop executes over a defined number of steps (typically 350), and performs the following operations at each time step:

a. Generates upstream concentrations.
b. Retrieves scheduled PID gains.
c. Calculates downstream emissions after lime neutralization.
d. Computes PID correction based on the deviation from setpoint.
e. Estimates the required lime dose.
f. Applies delay simulation using deque buffers to represent:
   - Measurement delay
   - Actuation delay
   - Prediction lag
g. Stores upstream, downstream, and lime dosing values for analysis.

The loop also logs detailed output for each step, facilitating controller debugging and parameter tuning.

7. Plotting Results

The script visualizes:

- Upstream SO_2 and HCl concentrations.
- Downstream simulated emissions with respect to setpoints.
- Lime dosing time series.

This allows the user to quickly assess:

- How closely the emissions track the setpoints.

- Whether the lime dosing remains within limits.
- The presence of overshoot, steady-state error, or instability.

4.2.3.3 Controller_Validation_RealData.py — Closed-Loop Simulation with Real Plant Data

This script serves as the **core validation module** for the lime dosing control strategy. Unlike `PID_Tunning_StepSim.py`, which uses synthetic step inputs, this script evaluates controller performance under **real operating conditions** by processing historical datasets from the plant.

It simulates how the feedforward–PID controller would have behaved if it had been active during the recorded data period and compares the simulated results to the actual lime dosing and downstream emissions recorded in the datasets. This enables both **technical validation** of the control logic and **quantitative performance benchmarking**.

General Workflow

The script processes multiple data files in batch. For each dataset, it:

1. Cleans and extracts relevant time series data.
2. Runs a time-based simulation using upstream emissions, flue gas flow, and flue gas composition.
3. Calculates controller response and estimates downstream SO_2 and HCl.
4. Compares simulation results with real measured values.
5. Aggregates and visualizes daily and monthly performance indicators.

Key Components and Functional Breakdown

1. Imports and Global Constants

- Standard libraries (`numpy, matplotlib, pandas`, etc.) are imported.
- Maximum and minimum allowable lime dosing rates are defined via:

```
MAX_SCREW = 1380 * 1
```

```
MIN_SCREW = 1380 * 0.09
```

These bounds are enforced later to simulate physical dosing system constraints.

2. PID Controller Class (`PIDController`)

This class encapsulates the PID algorithm logic:

- Accepts proportional (`Kp`), integral (`Ki`), and derivative (`Kd`) gains.
- Implements a `calculate()` method to compute the control signal.
- Tracks previous errors and includes an integral reset mechanism to manage windup.
- Allows dynamic gain updates using the `update_gains()` method.

One instance of this class is used per pollutant: SO_2 and HCl.

3. Gain Scheduling Function (`gain_schedule`)

A critical feature of this implementation is **gain-scheduling**, which allows the PID gains to change dynamically based on the current upstream concentrations of SO_2 and HCl. The logic is implemented via nested dictionaries and range comparisons.

This enhances performance across various operating regimes, making the controller adaptive to low or high pollutant loads.

4. Feedforward and Lime Dosing Calculations

Two functions handle dosing logic:

- **`feedforward_lime_dosing()`**:
  Calculates stoichiometric lime requirements for SO_2 and HCl based on deviation from setpoints and flue gas flow.

- **`lime_dosing_control()`**:
  Combines feedforward output with PID corrections. Each pollutant contributes separately to the final lime dosage:

```
lime_needed_SO2 = ...
```

```
lime_needed_HCL = ...
lime_needed = lime_needed_SO2 + lime_needed_HCL
```

Includes optional decoupling constants (set to zero in this case) to avoid cross-interference between loops.

Final lime output is clipped between `MIN_SCREW` and `MAX_SCREW`.

5. Simulation Engine (`simulate_pid_controller`)

This is the heart of the script. For each time step, it:

5.1. Retrieves upstream SO_2, HCl, airflow, H_2O, and O_2 levels from the dataset.

5.2. Applies correction factors for moisture and oxygen using:

```
cf_h2o = 100 / (100 - h2o_level)
cf_o2 = (20.95 - 11) / (20.95 - o2_level)
```

5.3. Applies gain scheduling to update the controller gains.

5.4. Calculates PID output and combines it with the feedforward estimation.

5.5. Converts lime dosing to pollutant neutralization using known efficiencies.

5.6. Subtracts neutralized quantities from upstream values to estimate downstream concentrations.

5.7. Applies correction factors to normalize emissions back to 11% O_2 and dry air.

5.8. Implements measurement delay simulation using a deque buffer:

```
downstream_SO2_d = deque([...], maxlen=delay)
...
downstream_SO2_d.append(downstream_SO2)
downstream_so2_values.append(downstream_SO2_d[0])
```

The full simulation captures not only the chemical effects but also the dynamic response of the control system to real process disturbances.

6. Dataset Processing Loop

The script processes multiple datasets using the following logic:

```
file_list = glob.glob("limeDosingControl_*.csv")
for file in file_list:
    ...
```

```
    simulate_pid_controller(...)
```

Each dataset corresponds to approximately one day of data. After processing:

- Daily averages are stored for SO_2, HCl, and lime dosing.
- Actual (real plant) values are recorded in parallel for comparison.
- Values are later aggregated to calculate monthly averages across all files.

7. Result Visualization and Output

The script generates two sets of plots:

- **Daily Time Series**: Shows upstream emissions, downstream simulation, and actual vs simulated lime dosing.
- **Monthly Averages**:
    - Shows comparisons of simulated vs. actual SO_2 and HCl averages.
    - Applies correction factors (e.g., scaling by 0.8 or 0.6) to print corrected averages.

Print statements also report daily and overall statistics:

```
print(f'Mean SO2 emissions: {avg_so2:.2f} vs Actual: {actual_avg_so2:.2f}')
```

4.2.3.4 Daily_ControllerVisualizer.py — Visual Performance Assessment on Daily Dataset

This script is a **focused analysis tool** designed to evaluate the lime dosing controller on a **single selected dataset**, typically representing a 24-hour operational period. Its primary purpose is to provide **clear visual insight** into the day-to-day performance of the controller under real conditions.

Unlike `Controller_Validation_RealData.py`, which performs batch processing and aggregates multi-day results, this script allows the user to interactively choose a specific file and inspect its time series in detail.

Structure and Functionality

The internal architecture of Daily_ControllerVisualizer.py largely reuses the control logic defined in Controller_Validation_RealData.py, including:

- The PIDController class for feedback correction, with gain scheduling.
- The stoichiometric feedforward module based on upstream SO_2 and HCl levels.
- Correction factors for flue gas moisture (H_2O) and oxygen (O_2).
- Neutralization efficiencies and decoupled contribution of SO_2 and HCl to lime demand.
- Delay simulation using deque structures.

The script avoids duplication by adopting the same formulae and functional logic, ensuring consistency across simulations.

Key Components and Functions

The internal architecture of Daily_ControllerVisualizer.py largely reuses the control logic defined in Controller_Validation_RealData.py, including:

- The PIDController class for feedback correction, with gain scheduling.
- The stoichiometric feedforward module based on upstream SO_2 and HCl levels.
- Correction factors for flue gas moisture (H_2O) and oxygen (O_2).
- Neutralization efficiencies and decoupled contribution of SO_2 and HCl to lime demand.
- Delay simulation using deque structures.

The script avoids duplication by adopting the same formulae and functional logic, ensuring consistency across simulations.

Workflow Summary

1. **User-Guided File Selection:**

The user is prompted to select a .csv dataset from a sorted list. This allows focused inspection of specific operational days.

2. **Data Preprocessing:**

The selected file is read and cleaned. Relevant time series – including upstream/downstream SO_2 and HCl, airflow, lime dosing, H_2O, and O_2 – are extracted and converted to NumPy arrays.

3. **Simulation Loop:**

For each timestep:

- Upstream emissions are processed.
- Feedforward and PID control actions are computed.
- Lime dosing is clipped to predefined physical limits.
- Downstream SO_2 and HCl are estimated using the same neutralization model and delay compensation methods as in the main validation script.
- All values are stored for plotting.

4. **Visualization:**

The script generates three plots:

- **SO_2 emissions**: Simulated vs. actual downstream concentrations.
- **HCl emissions**: Simulated vs. actual.
- **Lime dosing**: Simulated vs. recorded plant values.

5. **Daily Summary Metrics:**

Prints mean values for:

- Simulated and actual downstream SO_2 and HCl
- Simulated and actual lime injection

These metrics allow quick visual and numerical assessment of the controller's effectiveness on the selected day.

4.3 IMPLEMENTATION

The implementation of the designed system was a multi-step approach comprising the following key activities:

- **DCS Programming**
- **Development of a New Interface**
- **HAZOP Study**
- **Operations Manual Preparation**
- **System Testing, Monitoring and Tunning**

4.3.1 DCS PROGRAMMING

To deploy the new control strategy, the original Python algorithm was translated into the plant's native programming environment, specifically using FbCAD language. The logic was implemented in file `1HTK10DF003`. Notably, the **Smith Predictor** was intentionally excluded due to limited confidence in the process model's accuracy and the additional complexity its implementation would have introduced.

The schematics of the new control logic – developed in FbCAD – are presented in **APPENDIX VII** – DCS Control Loop. This section outlines the structure and function of the control modules across three pages:

Page 1/3 – Input acquisition, and Proportional and Integral Action

- **Input acquisition**

On the left side of the page, all relevant variables and constants – such as setpoints, upstream/downstream emissions, flue gas flow, molar masses, confidence factors, and PID gains – are defined. These are copied into reference blocks to instance them across the

program, promoting a clean and modular architecture for easier identification and maintenance.

- **Proportional Action**

This module calculates the action based on the instantaneous error between the corrected downstream emission and the setpoint. The error is multiplied by its corresponding proportional gain, and both the intermediate values (errors, gains, and actions) are assigned to output variables for monitoring.

- **Integral Action**

The integral term accumulates error over time. Since the DCS operates on a 1-second scan cycle, but downstream emissions update every 30 seconds, multiplexers were implemented to freeze the integrator unless a new downstream value is detected. These are controlled via the ***aux_XXX_2*** variables (explained under Anti-Windup). Additional multiplexers are used for manual reset operations via ***reset_XXX***, covered in the Integral Reset section. Once calculated, the integrals are multiplied by their respective gains to produce the integral actions, which are also made available as outputs.

Page 2/3 – Derivative Action, Control Factor Calculation, and Injection Limits

- **Derivative Action**

This module responds to changes in error by calculating its gradient. Comparator blocks detect differences between the current and previous error values, with ***prev_error_XXX*** updated accordingly. When new downstream values are available, the ***aux_XXX_1*** signal is activated to enable updates to the derivative (and integral) terms. After calculation, the derivatives are multiplied by the derivative gains to produce the derivative actions, also exposed as outputs.

- **Control Factor Calculation**

Here, the **control factor** (***cf_XXX***) is determined as the sum of the proportional, integral, and derivative actions for each pollutant.

- **Injection Limits**

As the system operates with two screws (main and standby), each with distinct operational ranges, this block defines the maximum and minimum allowable lime dosages (***MAX_LIME***, ***MIN_LIME***), which are also referenced in the Anti-Windup logic.

Page 3/3 – Lime injection calculation, Integral Reset Mechanism, and Anti-Windup Mechanism

- **Lime Injection Calculation**

The stoichiometric lime required for neutralizing SO_2 and HCl down to their setpoints is computed. Due to a measured system efficiency of ~32%, the actual required lime is approximately triple the stoichiometric value. To accommodate this, the lime dose is computed as indicated in **Equation 5**:

$$Lime_{Dosing} = stoich. \cdot (2 + cf_{XXX})$$

Equation 5 Lime Dosing Calculation.

This formulation ensures faster system response upon reactivation and enhances stability. The final lime injection is the sum of both pollutant-specific requirements, constrained within the established injection limits. Outputs include the stoichiometric values, correction factors, and individual/total lime injections.

- **Integral Reset Mechanism**

The logic governing ***reset_XXX*** variables is defined here. These triggers reset the integral term under two conditions

1. **Manual reset** by the operator after atypical operation (e.g., post-outage).
2. **Mode switch** from automatic to manual, preventing error accumulation while inactive.

- **Anti-Windup Mechanism**

To prevent excessive accumulation in the integral term – especially during prolonged deviations – an anti-windup system has been implemented. Unlike the legacy design, this uses a clamping logic based on ***aux_XXX_2*** which freezes the integral under the following conditions:

1. **No new downstream reading** (***aux_XXX_1 = 0***) – prevents stale error accumulation.
2. **Injection limits reached** – clamps the integral until error direction changes (e.g., positive error needed after reaching lower limit).
3. **Interaction safeguard** – prevents one controller's integral from reducing the total lime dosage when the other requires more. Specifically, the integral is only updated if:
   - The corresponding error is positive, or
   - Both errors are negative, and the correction factor is ≥ -1 (to avoid negative dosing suggestions).

4.3.2 DEVELOPMENT OF A NEW INTERFACE

Prior to the implementation of the new controller, the user interface (see **APPENDIX VIII** – Original Interface) was updated to reflect new information regarding the new controller and provide real-time insight into its performance. See **APPENDIX IX** – Modified Interface for detailed interface schematics.

Following the final tuning, a new interface was developed to offer enhanced visualisation of both the process' dynamics and the controller's internal features. This improved interface was also designed to restrict the operators' interaction with the new controller, limiting manual input to the adjustment of set points, and integral reset only. This measure aimed to prevent unintended disruptions and ensure consistent control behaviour.

The layout and functionalities of the final interface are presented in **APPENDIX X** – Final Interface.

4.3.3 HAZOP STUDY

To ensure a safe and reliable transition to the new lime dosing controller, a **Hazard and Operability (HAZOP) Study** [7] was conducted (see **Table 4** in **APPENDIX XI** – HAZOP Study). This structured analysis identified potential deviations from expected system behaviour, their underlying causes, possible consequences, and the protection measures implemented to mitigate associated risks.

The study focused on key areas of the process, including **pollutant emissions, lime dosing, controller maintenance**, and **operator interaction**, and used standard guide words such as *Over*, *Deviation*, and *Other than intended*.

Key findings and mitigation strategies included**:**

- **Pollutants – Overemission:**

One of the main risks identified was the potential for overemission due to incorrect controller tuning. This could result in regulatory breaches. To mitigate this, a **selector switch** was added to the DCS interface to allow operators to toggle between the legacy and the new controller. During the tuning phase, operation was supervised by the system developers. Additionally, **HCl and SO_2 alarms** were configured to activate at high-high (HH) levels, prompting operators to switch to manual mode and take corrective action.

- **Lime – Overdosing:**

Excessive lime injection due to poor tuning could lead to **baghouse blockages**. Protection is provided via **differential pressure sensors** which trigger an alarm when predefined thresholds are exceeded. In such cases, operators must manually access and clean the baghouse. Associated maintenance steps are outlined in **LN-02-01-OP-OPSBH-004-01-MS** and **LN-02-01-OP-OPSBH-001-01-MS**.

- **Maintenance – Tuning and Integral Windup:**

Risks associated with incorrect tuning or prolonged controller outages were also considered. To prevent **over- or underdosing**, the controller was implemented in **FB CAD**, making it

accessible to any trained DCS technician. A detailed **Operations Manual** is also available, providing clear instructions for gain modification within the **1HTK10DF003** loop. Additionally, **integral windup** caused by controller inactivity is addressed by two mechanisms: an **Integral Reset button** and **automatic reset** when switching to manual mode. These features prevent excessive error accumulation. All procedures are described in the **Lime Dosing System Operations Manual**.

- **Operations – Misuse or Misunderstanding:**

To reduce the risk of incorrect system operation, all operators have access to a **training document** explaining the new controller's functionality. This ensures consistent understanding and safe operation of the system.

Conclusion

The HAZOP study provided a robust framework to anticipate and address possible operational risks. As a result, the **commissioning process was completed successfully and without incident**, confirming that the safeguards and mitigation strategies in place were effective in ensuring a **safe and controlled start-up**.

4.3.4 OPERATIONS MANUAL PREPARATION

An operations manual has been developed to guide users through the functionality, configuration, and proper use of the new lime dosing control system. Its main objective is to ensure that operators, engineers, and maintenance personnel understand the key features and principles behind the system's design, particularly its transition from a semi-manual to a fully automated PID-based control strategy. The document includes a comparative analysis between the previous and current controllers, a detailed explanation of the new control logic and interface, and operational recommendations during the implementation phase. By providing a clear and structured reference, this manual aims to support safe operation, optimize performance, and facilitate informed decision-making during both normal operation and transitional stages such as tuning or troubleshooting.

All the above can be verified in detail throughout the contents of the file **Lime_Dosing_System_Operations_Manual.pdf**.

4.3.5 SYSTEM TESTING, MONITORING AND TUNNING

Once enough confidence had been gained in the initial gain configuration – specifically when the lime dosing proposed by the open-loop controller aligned coherently with that of the existing system – the selector switch was toggled, and the new system was brought online.

Prior to the first testing sessions, the initial configuration failed to deliver satisfactory performance. Significant adjustments were required: the **proportional gains (K_P)** were substantially increased to enable more effective responses to emission spikes and drops, while the **integral gains (K_I)** were reduced to prevent the excessive accumulation of past errors, which could compromise system stability and efficiency.

Additionally, the system demonstrated greater-than-expected stability across various operating points. As a result, the gain configuration was unified across conditions, effectively simplifying the control architecture to a **conventional PID** rather than a gain-scheduled structure. This change aimed to identify a stable and robust gain set that yielded acceptable results across a wide range of scenarios.

The final tuning parameters for each pollutant-specific PID loop are summarised in **Table 1** and **Table 2**:

HCl Gain Set Up	
Kp	1.15
Ki	0.03
Kd	0

Table 1 HCl PID Gain Set Up.

SO2 Gain Set Up	
Kp	1.5
Ki	0.01
Kd	0

Table 2 SO2 PID Gain Set Up.

Justification of the Final Configuration

Before detailing the reasoning behind each selected gain, it is important to consider the distribution of lime usage between the two pollutant control loops: approximately **6/7 of the total lime dosing** is consumed in **HCl neutralisation**, while only **1/7** is used for **SO_2**. Therefore, the HCl controller largely determines the base lime dosing. Nevertheless, due to the significantly higher **standard deviation (STD)** observed in SO_2 downstream emissions (see **APPENDIX XII** – Final Report), the SO_2 loop requires higher reactivity to compensate for its smaller contribution to the total dose.

1. **Proportional Action**

The initial **HCl K_P** was set up to 1.00 as it yielded lime values comparable to the previous controller. After closing the loop and configuring the integral gain (K_I), it was increased to **1.15**. This adjustment was made to reduce lime usage after prolonged periods of over-emission – when the integral action becomes dominant – by enhancing the controller's responsiveness to negative deviations, thereby improving overall efficiency.

In contrast, the **SO_2** loop's **K_P** was set to **1.50**, representing a **50% increase** over the HCl gain. This value provides sufficient weight in the total lime dosing calculation to address SO_2 emission spikes, while minimizing interference with HCl control and avoiding actuator over-response.

2. **Integral Action**

The initial **K_I** was set to **0.01** in both loops, allowing fast correction of SO_2 deviations and good disturbance rejection. However, this configuration led to persistent errors in HCl

emissions. Therefore, the **HCl** K_I was increased to **0.03**, improving steady-state tracking without introducing excessive oscillations.

3. Derivative Action

Although derivative action (K_D) is typically used to dampen oscillations by reacting to the error's rate of change, its implementation was deemed counterproductive in this application. The system presents a **~2-minute delay** in both actuator response and downstream measurement, meaning that the derivative term would compute a gradient from the past (approx. 2 minutes earlier) to correct a future event (approx. 2 minutes ahead). This mismatch would likely deteriorate system performance. As a result, the derivative action was **disabled** in both controllers by setting $K_D = 0$.

Section 5. RESULTS ANALYSIS

To evaluate and contrast the performance of the original lime dosing controller with that of the newly implemented system, a series of 24-hour reports were compiled and analysed. These reports consisted of process data sampled at 40-second intervals, enabling a detailed assessment of both efficiency and emission control performance.

The dataset included the following key process variables:

- Lime dosing. [kg/h]
- Upstream HCl concentrations. [mg/Nm³]
- Upstream SO_2 concentrations. [mg/Nm³]
- Downstream HCl concentrations. [mg/Nm³]
- Downstream SO_2 concentrations. [mg/Nm³]
- Flue gas flow rate. [Nm³/h]
- Total lime consumption. [kg]
- Incinerated waste. [t]

The upstream pollutant concentrations and the flue gas flow were used to calculate the stoichiometric lime demand required to neutralise SO_2 and HCl emissions down to their respective set points. Based on this, the **lime injection-to-stoichiometric ratio** was determined as a **key indicator of dosing efficiency**. Additionally, the **deviations of downstream emissions from their regulatory set points** were analysed to **assess overall control accuracy and compliance**.

Finally, the **Lime-to-Waste ratio** was also considered in the performance and economic analysis. This parameter is obtained by dividing the total daily lime consumption [kg] by the daily incinerated waste [t].

The evaluation involved two levels of analysis:

1. **Day-to-Day Comparison**

A direct comparison was conducted between the 24-hour performance of the new controller and the performance of the original system on the same calendar day from the previous year. This approach ensured seasonal and operational parity between the datasets.

2. **Long-Term Efficiency Assessment**

The average dosing efficiency of the legacy controller over a **one-year period** was compared to the performance of the new controller during its operational timeframe. This allowed for an aggregated evaluation of the new system's performance consistency and its potential for long-term lime consumption optimisation.

5.1 KEY FINDINGS

The results presented in **APPENDIX XII** – Final Report support the following conclusions:

1. **Lime Consumption Reduction**

Both performance assessments revealed a consistent reduction in lime consumption following the implementation of the new control strategy:

- The **day-to-day comparison** revealed a **14% decrease** in **lime consumption**.
- The **long-term projection** also indicates a **potential reduction of up to 14%**, depending on operational consistency and load conditions.

While these figures are promising, it is important to note that **lime consumption alone does not provide a fully reliable measure of controller performance**, as it is strongly influenced by fluctuating operating conditions. Therefore, this analysis must be considered in conjunction with the **efficiency** and **accuracy** studies presented in the following sections.

2. **Improved Efficiency**

Both performance assessments demonstrate a clear improvement in lime efficiency following the implementation of the new controller.

- The **day-to-day comparison** revealed a **31% decrease** in the **lime consumption-to-stoichiometric ratio**.
- The **long-term projection** indicates a **potential reduction of up to 35%**, depending on operational consistency and load conditions.

These results confirm that the project has successfully met – and significantly surpassed – the original 5% reduction target for lime consumption.

3. Enhanced Accuracy and Regulatory Compliance

Prior to the deployment of the new control system, the **day-to-day comparison** showed that the plant struggled to maintain emissions below setpoints without relying on confidence factors. This is evidenced by **average positive deviations** of **+28.6% for HCl**. However, the **long-term evaluation** indicates an improvement in tracking performance, with average deviations of **-9.5% for SO_2** and **+2.3% for HCl**.

In contrast, the **new controller** achieves a **-1.3% deviation in HCl emissions** – the dominant variable in lime dosing. This reflects a **higher degree of precision** and more consistent **regulatory compliance**, achieved **without relying on correction margins**. This reliability accredited the new controller to feed back the downstream values with confidence factors applied, which resulted in the achievement of higher efficiencies.

It is worth noting, however, that the tracking of the SO_2 setpoint appears to have **worsened** under the new system. This is **not considered a concern**, for the following reasons:

1) **Higher volatility**: SO_2 naturally exhibits greater variability, and its setpoint tracking becomes more erratic – especially when deviations are negative.
2) **Gas interaction effects**: Lime neutralisation is inherently interdependent between SO_2 and HCl, and the reactions are nonlinear and cross-coupled. As a result, the controller tends to prioritise minimising the dominant gas' deviation, often at the expense of the secondary one.
3) **Post-maintenance conditions**: At the time of the controller's commissioning, the plant had recently undergone a major outage, during which several components were renewed

or optimally maintained. This improved **heat transfer conditions** in the flue gas path. Since **SO_2 neutralisation is strongly temperature-dependent**, the improved heat exchange likely reduced SO_2 emissions temporarily. As equipment performance gradually degrades, **SO_2 dosing requirements are expected to increase**, bringing its average deviation closer to the setpoint again.

In addition to the above, although the **standard deviation** of SO_2 and HCl emissions **decreased** with the new controller – indicating better **absolute control stability** – the **coefficient of variation (Cv)** for both gases showed a **slight increase**. This apparent contradiction stems from the significant drop in average emission levels: **a smaller absolute spread becomes proportionally larger when the mean is lower**.

This **does not indicate a deterioration in control quality**. On the contrary, the combined reduction in both mean values and standard deviations confirms that the new controller provides **more accurate, efficient, and stable performance** – even when operating **closer to regulatory limits**. The system therefore achieves **greater overall robustness and optimisation**, both environmentally and economically.

4. Reduction of Lime-to-Waste Ratio

The Lime-to-Waste Ratio compares the daily lime consumption against the daily mass of waste incinerated. The Final Report exhibits a **10% reduction**, indicating an **improvement in efficiency** – less lime is required per kg of waste processed.

While the outcome of this evaluation is positive, it does not reveal a categorical conclusion regarding the system's performance. This is because this parameter is highly dependent on the composition of the waste feed. Certain materials, such as plastics, generate higher concentrations of pollutants – particularly HCl when combusted.

For this reason, although the observed reduction in Lime-to-Waste Ratio is noteworthy, it was not considered a decisive factor in evaluating the overall success of the project.

5.2 *ECONOMIC EVALUATION*

Before conducting the economic evaluation, it is important to clarify that all cost estimates are referenced to this facility's **annual lime expenditure**, which is approximately **£400,000**, based on a unit price **£200/ton**.

Given that lime consumption is influenced by several variables – such as the quantity and composition of incinerated waste, as well as the environmental conditions like temperature – this analysis is structured in 3 distinct approaches. The outcome of each will be employed to provide comprehensive cost-saving estimation.

First Approach – Lime-to-Stoichiometric Lime Ratio (Lime2Sto) Analysis

To ensure a fair and normalised comparison of system performance, the analysis focused on days within the controller's operational period where **daily average stoichiometric lime requirement** fell within the observed minimum and maximum bounds. This filtering approach provides a consistent basis for evaluating controller efficiency, as the stoichiometric lime value reflects the underlying pollutant load.

The **filtered average Lime2Sto** ratio under the previous controller was found to be **3.20**, compared to the **2.20** achieved by the new system. This equates to a **31% improvement in efficiency**. Assuming this gain leads to a proportional reduction in lime consumption, the estimated annual cost saving is **£124,000**.

Second Approach – Lime Consumption Analysis

Applying the same filtering method, the **average filtered daily lime injection** under the old controller was **260.2 kg/h**, while under the new controller it dropped to **206.4 kg/h**. This represents a **21% reduction in lime usage**, corresponding to an estimated **annual saving of £84,000**.

Third Approach – Lime-to-Waste Ratio

As previously discussed, the **Lime-to-Waste Ratio assessment** – which relates lime consumption to the quantity of waste incinerated – indicates a **10% reduction** following the implementation of the new control system. This translates into an estimated saving of **£40,000** per year.

Conclusion

While it is difficult to precisely forecast the total cost savings of this Project – due to various influencing factors such as operational variability and potential lime price increases – the results suggest that the **new control strategy could deliver annual savings between £40,000 and £124,000**, depending on operating conditions.

Assuming a **useful life of 5 years**, after which the system may be subject to revision due to potential technological advancements and considering a conservative scenario with minimum projected savings (£40,000/year) and a **2% annual inflation in lime price**, the Project demonstrates strong financial viability. Based on these assumptions, the following economic indicators were obtained:

- **Return on Investment (ROI):** 61
- **Internal Rate of Return (IRR):** 1,186%
- **Payback Period:** ~1 month

These results confirm that the Project offers **exceptional profitability with minimal financial risk**, even under conservative projections.

For a detailed breakdown of the assumptions and calculations, refer to **Table 5** in **APPENDIX XIII – Economic Study**.

Section 6. CONCLUSIONS AND FUTURE PROJECTS

The primary objectives of this project were to:

1. Maintain SO_2 and HCl emissions below regulatory limits,
2. Automate the lime dosing process.
3. Reduce lime consumption by at least 5%.

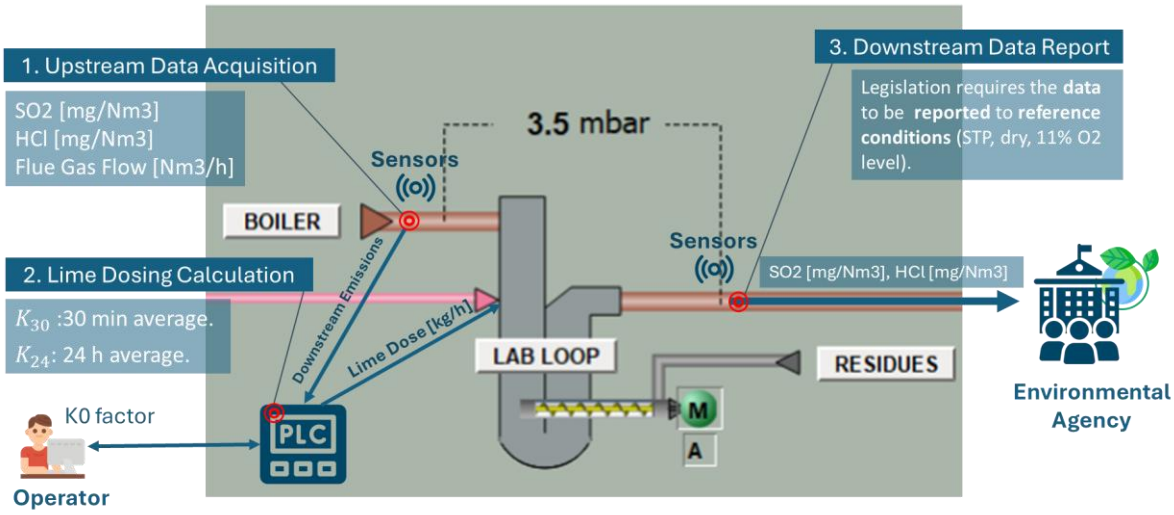
The implementation of the new PID-based control system has successfully fulfilled all three objectives. The system demonstrated a **significant improvement in emission control accuracy**, ensuring compliance without relying on external confidence factors. The newly developed interface enables **full automation** of the dosing process, **reducing operator intervention** and **standardizing system performance**. Additionally, both short-term and long-term evaluations revealed considerable reductions in lime usage – surpassing the initial 5% target and indicating **potential savings of up to 31%** depending on operational conditions.

Looking ahead, **future developments** could focus on **extending this control strategy to other energy-from-waste (EfW) facilities**, leveraging the adaptability of the control logic and interface. Furthermore, integrating **AI-based prediction models** could enhance system performance by mitigating the effects of process delays, particularly those related to actuator response and downstream emission readings. These enhancements would further optimise reagent usage and improve the responsiveness of the system under variable plant conditions.

Section 7. BIBLIOGRAPHY

[1] «Energy from waste: a guide to the debate», GOV.UK. Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.gov.uk/government/publications/energy-from-waste-a-guide-to-the-debate
[2] «Homepage - FCC Environment». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.fccenvironment.co.uk/
[3] «Valmet DNA Distributed Control System». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.valmet.com/automation/control-systems/dna/
[4] «Control and Safety Systems | Emerson US». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.emerson.com/en-us/automation/control-and-safety-systems
[5] P. N. Chisholm y G. T. Rochelle, «Dry Absorption of HCL and SO2 with Hydrated Lime from Humidified Flue Gas», *Ind. Eng. Chem. Res.*, vol. 38, n.º 10, p. 4068, 1999, doi: 10.1021/IE9806601.
[6] «Control of Processes with Long Dead Time: The Smith Predictor - MATLAB & Simulink Example». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://es.mathworks.com/help/control/ug/control-of-processes-with-long-dead-time-the-smith-predictor.html
[7] «HAZOP Study | Process Safety Training - Courses - IChemE». Accedido: 11 de julio de 2025. [En línea]. Disponible en: https://www.icheme.org/training-events/training/courses-a-z/hazop-study-for-team-leaders-and-team-members/

APPENDIX I – PROCESS SCHEME



APPENDIX II – PROJECT PLANIFICATION

SUMMARY TABLE

Stage	Activity	Description	Resources	Duration	Cost
System Model	Data Gathering and Analysis	-Extract historical data from the plant's Decentralized Control System (DCS). -Identify relevant parameters and understand their correlations.	Excel & DNA Operate Client	1 week	£ 262.50
	Process Recreation in Python	- Develop a script in Python to replicate the process' behaviour.	Visual Studio	3 weeks	£ 350.00
Controller Design	Controller Design & Fine Tunning	- Choose an adequate control strategy for the process. - Fine tune the controller. - Analyse the simulations' reports.	Visual Studio	6 weeks	£1,575.00
Implementation	Python to Native Language Translation	- Translate the original Python algorithm into FbCAD and Sequence CAD	DNA Explorer	1 day	£ 35.00
	New Interface Development	- Develop a new interface for the new controller.	DNA Explorer	1 day	£ 35.00
	HAZOP Study	- Conduct a HAZOP study.	Excel	1 day	£ 35.00
	Operations Manual Preparation	- Write a comprehensive manual to guide users.	Word	1 day	£ 35.00
	System Testing, Monitoring & Tunning	- Evaluate the system's performance & carry out adjustments/changes.	DNA Explorer	4 weeks	£ 1,050.00
			TOTAL	**14 weeks & 4 days**	**£3,377.50**

Table 3 Project Planification Summary.

APPENDIX III – EFFICIENCY_ESTIMATION.PY

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import glob
from natsort import natsorted
from collections import deque

# Constants
MAX_SCREW = 1380 * 1  # [kg/h]
MIN_SCREW = 1380 * 0.09  # [kg/h]

# Get list of available datasets
file_list = glob.glob("limeDosingControl_*.csv")
file_list = natsorted(file_list)
print("Available datasets:")
for i, file in enumerate(file_list):
    print(f"{i+1}: {file}")

# Allow user to choose a dataset
dataset_index = int(input("Select dataset number: ")) - 1
selected_file = file_list[dataset_index]

# Load the CSV file
data = pd.read_csv(selected_file, low_memory=False)

# List of column names to check
columns = ['SO2 Upstream', 'HCl Upstream', 'AirFlow', 'Lime Injected', 'H2O', 'O2
dry', 'SO2 Downstream corrected', 'HCl Downstream corrected']

# Select the range
data_subset = data[columns][3:8640]

# Clean Data
clean_data = data_subset.replace(['#REF!', 'NaN', '', None],
float('nan')).dropna()

# Find rows with NaN values
invalid_rows = clean_data.isna().any(axis=1)

# Keep only valid rows
clean_data = clean_data[~invalid_rows]

# Convert each cleaned column to a NumPy array
so2_values_us = clean_data['SO2 Upstream'].astype(float).values
fg_flow_values = clean_data['AirFlow'].astype(float).values
actual_lime_dosing = clean_data['Lime Injected'].astype(float).values
h2o_values = clean_data['H2O'].astype(float).values
```

```
o2_values = clean_data['O2 dry'].astype(float).values
so2_values_ds = clean_data['SO2 Downstream corrected'].astype(float).values
mean_so2_ds = np.mean(so2_values_ds)

steps = 8600
efficiency_so2 = 0.04
downstream_so2_values = []
upstream_so2_values = []
lime_dosing = []
delay_reading = 6
delay_act = 1
downstream_SO2_d = deque([0] * delay_reading, maxlen=delay_reading)
downstream_HCL_d = deque([0] * delay_reading, maxlen=delay_reading)
lime_dosing_d = deque([0] * delay_act, maxlen=delay_act)

# Simulation:
for step in range(steps):
    # Get values from dataset
    upstream_SO2 = so2_values_us[step]
    fg_flow = fg_flow_values[step]
    h2o_level = h2o_values[step]
    o2_level = o2_values[step]

    # Calculate correcting factors for upstream emissions
    cf_h2o = 100/(100-h2o_level)
    cf_o2 = (20.95-11)/(20.95-o2_level)

    # Transform upstream values to kg/h
    upstream_SO2_kg_h = upstream_SO2* fg_flow/10**6

    # Calculate neutralised pollutants [kg/h]
    neutralized_SO2 = max(actual_lime_dosing[step] * 64 / 74 * efficiency_so2,0)

    # Calculate downstream values [kg/h]
    downstream_SO2_kg_h = max(upstream_SO2_kg_h - neutralized_SO2,0)

    # Transform downstream values to mg/Nm3
    downstream_SO2 = downstream_SO2_kg_h / (fg_flow/10**6)

    #Apply correcting factors
    downstream_SO2 /= (cf_h2o*cf_o2)

    downstream_SO2_d.append(downstream_SO2)
    lime_dosing_d.append(actual_lime_dosing[step])

    # Append values for plotting
    downstream_so2_values.append(downstream_SO2_d[0])
    lime_dosing.append(lime_dosing_d[0])

# Plottings
lim_l = 3000
lim_h = 4000
```

```
downstream_so2_values = downstream_so2_values[lim_l:lim_h]
so2_values_ds = so2_values_ds[lim_l:lim_h]
lime_dosing = lime_dosing[lim_l:lim_h]

plt.figure(figsize=(10, 6))

plt.plot(downstream_so2_values, label="Estimated SO2", color='r')
plt.plot(so2_values_ds, label="Real SO2", color='b')
plt.title("Downstream SO2")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

APPENDIX IV – PID_TUNNING_STEPSIM.PY

```
import numpy as np
import matplotlib.pyplot as plt
from collections import deque

# Constants
MAX_SCREW = 1380 * 1  # [kg/h]
MIN_SCREW = 1380 * 0.09  # [kg/h]

class PIDController:
    def __init__(self, Kp, Ki, Kd, setpoint, integral_limit):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.setpoint = setpoint
        self.prev_error = 0
        self.integral = 0
        self.integral_limit = integral_limit

    def update_gains(self, Kp, Ki, Kd):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd

    def calculate(self, current_value, reset_integral):
        error = (current_value - self.setpoint)/self.setpoint
        P = self.Kp * error

        self.integral += error
        if reset_integral == 1: self.integral = 0
        I = self.Ki * self.integral

        D = self.Kd * (error - self.prev_error)
        self.prev_error = error

        output = P + I + D
        return output

    def get_integral(self):
        return self.integral

def simulate_steps(step):
    upstream_so2 = initial_SO2
    upstream_hcl = initial_HCL

    if step >= 15:
        upstream_so2 = 100
        upstream_hcl = 330
```

```
    if step >= 30:
        upstream_hcl = 450

    if step >= 45:
        upstream_hcl = 550

    if step >= 60:
        upstream_hcl = 650

    if step >= 75:
        upstream_hcl = 750

    if step >= 90:
        upstream_hcl = 850

    if step >= 105:
        upstream_so2 = 160
        upstream_hcl = 200

    if step >= 120:
        upstream_hcl = 450

    if step >= 135:
        upstream_hcl = 550

    if step >= 150:
        upstream_hcl = 650

    if step >= 165:
        upstream_hcl = 750

    if step >= 180:
        upstream_hcl = 850

    if step >= 195:
        upstream_so2 = 300
        upstream_hcl = 200

    if step >= 210:
        upstream_hcl = 450

    if step >= 225:
        upstream_hcl = 550

    if step >= 240:
        upstream_hcl = 650

    if step >= 255:
        upstream_hcl = 750

    if step >= 270:
        upstream_hcl = 850
```

```
    if step >= 285:
        upstream_so2 = 360
        upstream_hcl = 200

    if step >= 300:
        upstream_hcl = 450

    if step >= 315:
        upstream_hcl = 550

    if step >= 330:
        upstream_hcl = 650

    if step >= 345:
        upstream_hcl = 750

    if step >= 360:
        upstream_hcl = 850


    return upstream_so2, upstream_hcl

# Feedforward Lime Dosing Calculation
def feedforward_lime_dosing(fg_flow, efficiency_SO2, efficiency_HCl,
upstream_SO2, upstream_HCl, setpoint_SO2, setpoint_HCl):
    QSO2 = (upstream_SO2 - setpoint_SO2) * fg_flow/10**6  # [kg/h]
    QSO2 = max(0, QSO2)
    QHCL = (upstream_HCl - setpoint_HCl) * fg_flow/10**6 # [kg/h]
    QHCL = max(0, QHCL)

    return QSO2, QHCL

def gain_schedule(upstream_SO2, upstream_HCL):
    so2_ranges = [
        (120, {
            500: (1, 10, 0.05),
            #500: (1, 10, 0.05),
            float('inf'): (0.01, 0, 0.05)
        }),
        (180, {
            500: (1, 6, 0.05),
            float('inf'): (0.01, 10, 0.05)
        }),
        (250, {
            float('inf'): (0.001, 4, 0.05)
        }),
        (350, {
            float('inf'): (0.001, 2, 0)
        }),
        (float('inf'), {
            float('inf'): (0.001, 2, 0)
        })
```

```
    ]

    hcl_ranges = {
        280: (0.01, 0.15, 0.0),
        340: (0.025, 0.1, 0.0),
        500: (0.002, 0.05, 0.005),
        600: (0.001, 0.045, 0.0),
        700: (0.001, 0.035, 0.0),
        float('inf'): (0.001, 0.02, 0)
    }

    def find_gains(value, ranges):
        for limit, gains in ranges:
            if value < limit:
                return gains if isinstance(gains, tuple) else
find_gains(upstream_HCL, gains.items())

    Kp_SO2, Ki_SO2, Kd_SO2 = find_gains(upstream_SO2, so2_ranges)
    Kp_HCl, Ki_HCl, Kd_HCl = find_gains(upstream_HCL, hcl_ranges.items())

    return Kp_SO2, Ki_SO2, Kd_SO2, Kp_HCl, Ki_HCl, Kd_HCl

# Lime dosing calculation with Feedforward and PID
def lime_dosing_control(upstream_SO2, upstream_HCl, downstream_SO2,
downstream_HCl, setpoint_SO2, setpoint_HCl, fg_flow, efficiency_SO2,
efficiency_HCl, cf_SO2, cf_HCL, lime_residue):
    C_SO2 = 74/64*0
    C_HCL = 74 / (36.45 * 2) *0
    QSO2, QHCL = feedforward_lime_dosing(fg_flow, efficiency_SO2, efficiency_HCl,
upstream_SO2, upstream_HCl, setpoint_SO2, setpoint_HCl)
    lime_needed_SO2 = (QSO2 / 64) * 74 *(1+cf_SO2-C_HCL*cf_HCL)
    lime_needed_HCL = (QHCL / (36.45 * 2)) * 74 *(1 + cf_HCL-C_SO2*cf_SO2)
    lime_needed = lime_needed_SO2 + lime_needed_HCL
    return np.clip(lime_needed, MIN_SCREW, MAX_SCREW)

# Simulation function
def simulate_pid_controller(steps, setpoint_SO2, setpoint_HCl, initial_SO2,
initial_HCL, integral_limit):
    # PID controllers
    Kp_SO2, Ki_SO2, Kd_SO2 = 0.01, 15, 0.05
    Kp_HCl, Ki_HCl, Kd_HCl = 0.015, 0.15, 0.01
    pid_SO2 = PIDController(Kp_SO2, Ki_SO2, Kd_SO2, setpoint_SO2, integral_limit)
    pid_HCl = PIDController(Kp_HCl, Ki_HCl, Kd_HCl, setpoint_HCl, integral_limit)

    # Constants
    fg_flow = 120000 # [kg/h]
    efficiency_so2 = 0.04
    efficiency_hcl = 0.32

    # Plotted variables
    so2_values = [initial_SO2]
    hcl_values = [initial_HCL]
    upstream_so2_values = [initial_SO2]
```

```
    upstream_hcl_values = [initial_HCL]
    lime_doses = []

    # Other variables
    downstream_SO2 = initial_SO2 # [mg/Nm3]
    downstream_HCL = initial_HCL # [mg/Nm3]
    upstream_SO2 = initial_SO2
    upstream_HCL = initial_HCL
    cf_SO2 = 0
    cf_HCl = 0
    lime_residue = 0 # [kg/h]
    prev_Ki_SO2 = Ki_SO2
    prev_Ki_HCl = Ki_HCl
    reset_integral_SO2 = 0
    reset_integral_HCl = 0
    integral_so2 = 0
    integral_hcl = 0
    delay_reading = 6
    delay_estimate = 6
    delay_act = 12
    downstream_SO2_rd = deque([0] * delay_reading, maxlen=delay_reading)
    downstream_HCL_rd = deque([0] * delay_reading, maxlen=delay_reading)
    downstream_SO2_d = deque([0] * delay_reading, maxlen=delay_estimate)
    downstream_HCL_d = deque([0] * delay_reading, maxlen=delay_estimate)
    correction_SO2 = 0
    correction_HCL = 0

    #for step in range(len(so2_values_us)):
    for step in range(steps):
        # Simulate steps
        upstream_SO2, upstream_HCL = simulate_steps(step)
        Kp_SO2, Ki_SO2, Kd_SO2, Kp_HCl, Ki_HCl, Kd_HCl =
gain_schedule(upstream_SO2, upstream_HCL)
        pid_SO2.update_gains(Kp_SO2, Ki_SO2, Kd_SO2)
        pid_HCl.update_gains(Kp_HCl, Ki_HCl, Kd_HCl)

        # Reset integrals if needed
        integral_so2 = pid_SO2.get_integral()
        integral_hcl = pid_HCl.get_integral()
        if (integral_so2 < 0 and integral_hcl > 0): reset_integral_SO2 = 1
        else: reset_integral_SO2 = 0
        if (integral_so2 > 0 and integral_hcl < 0): reset_integral_HCl = 1
        else: reset_integral_HCl = 0

        # Transform upstream values to kg/h
        upstream_HCL_kg_h = upstream_HCL* fg_flow/10**6
        upstream_SO2_kg_h = upstream_SO2* fg_flow/10**6

        # Calculate correcting factors without delay
        cf_SO2 = pid_SO2.calculate(downstream_SO2+correction_SO2,
reset_integral_SO2)
        cf_HCl = pid_HCl.calculate(downstream_HCL+correction_HCL,
reset_integral_HCl)
```

```
        # Calculate lime dosing [kg/h]
        lime_dosing = lime_dosing_control(upstream_SO2, upstream_HCL,
downstream_SO2, downstream_HCL, setpoint_SO2, setpoint_HCl,
                                          fg_flow, efficiency_so2,
efficiency_hcl, cf_SO2, cf_HCl, lime_residue)
        lime_doses.append(lime_dosing)

        # Calculate neutralised pollutants [kg/h]
        neutralized_SO2 = max((lime_dosing * 64) / 74 * efficiency_so2 , 0)
        neutralized_HCL = max((lime_dosing * (36.45 * 2)) / 74 *
efficiency_hcl,0)

        # Calculate downstream values [kg/h]
        downstream_SO2_kg_h = max(0, upstream_SO2_kg_h - neutralized_SO2)
        downstream_HCL_kg_h = max(0, upstream_HCL_kg_h - neutralized_HCL)

        # Transform downstream values to mg/Nm3
        downstream_SO2 = downstream_SO2_kg_h / (fg_flow/10**6)
        downstream_HCL = downstream_HCL_kg_h / (fg_flow/10**6)

        downstream_SO2_d.append(downstream_SO2)
        downstream_HCL_d.append(downstream_HCL)
        downstream_SO2_rd.append(downstream_SO2)
        downstream_HCL_rd.append(downstream_HCL)
        correction_SO2 = downstream_SO2_rd[0] - downstream_SO2_d[0]
        correction_HCL = downstream_HCL_rd[0] - downstream_HCL_d[0]


        so2_values.append(downstream_SO2_rd[0])
        hcl_values.append(downstream_HCL_rd[0])
        upstream_so2_values.append(upstream_SO2)
        upstream_hcl_values.append(upstream_HCL)

    return so2_values, hcl_values, lime_doses, upstream_so2_values,
upstream_hcl_values

# Example usage
setpoint_SO2 = 40  # [mg/Nm3]
setpoint_HCl = 8.0  # [mg/Nm3]
initial_SO2 = 100  # [mg/Nm3]
initial_HCL = 200  # [mg/Nm3]
integral_limit = 1000
steps = 350
lim_l = 0
lim_h = 350

so2_values, hcl_values, lime_doses, upstream_so2_values, upstream_hcl_values =
simulate_pid_controller(steps, setpoint_SO2, setpoint_HCl, initial_SO2,
initial_HCL, integral_limit)


# Plot results
```

```
so2_values = so2_values[lim_l:lim_h]
hcl_values = hcl_values[lim_l:lim_h]
lime_doses = lime_doses[lim_l:lim_h]
upstream_so2_values = upstream_so2_values[lim_l:lim_h]
upstream_hcl_values = upstream_hcl_values[lim_l:lim_h]
plt.figure(figsize=(10, 6))

# Upstream values [mg/Nm3]
plt.subplot(3, 1, 1)
plt.plot(upstream_so2_values, label="Upstream SO2", color='r')
plt.plot(upstream_hcl_values, label="Upstream HCl", color='b')
plt.title("Upstream SO2 & HCl Concentration")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

# Downstream values [mg/Nm3]
plt.subplot(3, 1, 2)
plt.plot(so2_values, label="SO2 Concentration", color='r')
plt.plot(hcl_values, label="HCl Concentration", color='b')
plt.axhline(y=setpoint_SO2, color='r', linestyle='--', label="Setpoint SO2")
plt.axhline(y=setpoint_HCl, color='b', linestyle='--', label="Setpoint HCl")
plt.title("Downstream SO2 & HCl Concentration")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

# Lime doses [kg/h]
plt.subplot(3, 1, 3)
plt.plot(lime_doses, label="Lime Dosing", color='green')
plt.title("Lime Dosing Output")
plt.xlabel("Time Step")
plt.ylabel("Lime Dosing [kg/h]")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

APPENDIX V –

CONTROLLER_VALIDATION_REALDATA.PY

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import glob
from natsort import natsorted
from collections import deque

# Constants
MAX_SCREW = 1380 * 1  # [kg/h]
MIN_SCREW = 1380 * 0.09  # [kg/h]
avg_so2 = 0
avg_hcl = 0
avg_lime = 0
actual_avg_so2 = 0
actual_avg_hcl = 0
actual_avg_lime = 0
v_avg_so2 = []
v_avg_hcl = []
v_actual_avg_so2 = []
v_actual_avg_hcl = []

# Get list of available datasets
file_list = glob.glob("limeDosingControl_*.csv")
file_list = natsorted(file_list)
print(f"Processing {len(file_list)} datasets...")

class PIDController:
    def __init__(self, Kp, Ki, Kd, setpoint):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.setpoint = setpoint
        self.prev_error = 0
        self.integral = 0

    def update_gains(self, Kp, Ki, Kd):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd

    def calculate(self, current_value, reset_integral):
        error = (current_value - self.setpoint)/self.setpoint
        P = self.Kp * error
```

```
        if reset_integral == 1:
                self.integral = 0
        else: self.integral += error
        I = self.Ki * self.integral

        D = self.Kd * (error - self.prev_error)
        self.prev_error = error

        output = P + I + D
        return output

    def get_integral(self):
        return self.integral

def gain_schedule(upstream_SO2, upstream_HCL):
    so2_ranges = [
        (120, {
            500: (1, 10, 0.0),
            600: (1, 5, 0.0),
            float('inf'): (1, 0, 0.0)
        }),
        (180, {
            500: (1, 6, 0.0),
            float('inf'): (0.01, 10, 0.0)
        }),
        (250, {
            float('inf'): (0.001, 10, 0.0)
        }),
        (350, {
            float('inf'): (0.001, 6, 0)
        }),
        (float('inf'), {
            float('inf'): (0.001, 4, 0)
        })
    ]

    hcl_ranges = {
        280: (0.01, 0.15, 0.0),
        340: (0.025, 0.1, 0.0),
        500: (0.002, 0.05, 0.005),
        600: (0.001, 0.045, 0.0),
        700: (0.001, 0.035, 0.0),
        float('inf'): (0.001, 0.02, 0)
    }

    def find_gains(value, ranges):
        for limit, gains in ranges:
            if value < limit:
                return gains if isinstance(gains, tuple) else
find_gains(upstream_HCL, gains.items())

    Kp_SO2, Ki_SO2, Kd_SO2 = find_gains(upstream_SO2, so2_ranges)
```

```
    Kp_HCl, Ki_HCl, Kd_HCl = find_gains(upstream_HCL, hcl_ranges.items())

    return Kp_SO2, Ki_SO2, Kd_SO2, Kp_HCl, Ki_HCl, Kd_HCl

# Feedforward Lime Dosing Calculation
def feedforward_lime_dosing(fg_flow, upstream_SO2, upstream_HCl, setpoint_SO2,
setpoint_HCl):
    QSO2 = (upstream_SO2 - setpoint_SO2) * fg_flow/10**6  # [kg/h]
    QSO2 = max(0, QSO2)
    QHCL = (upstream_HCl - setpoint_HCl) * fg_flow/10**6 # [kg/h]
    QHCL = max(0, QHCL)

    return QSO2, QHCL

# Lime dosing calculation with Feedforward and PID
def lime_dosing_control(upstream_SO2, upstream_HCl,setpoint_SO2, setpoint_HCl,
fg_flow, cf_SO2, cf_HCL):
    # Lime calculation
    QSO2, QHCL = feedforward_lime_dosing(fg_flow, upstream_SO2, upstream_HCl,
setpoint_SO2, setpoint_HCl)
    lime_needed_SO2 = (QSO2 / 64) * 74 *(1+cf_SO2)
    lime_needed_HCL = (QHCL / (36.45 * 2)) * 74 *(1 + cf_HCL)
    lime_needed = lime_needed_SO2 + lime_needed_HCL

    return np.clip(lime_needed, MIN_SCREW, MAX_SCREW)

# Simulation function
def simulate_pid_controller(steps, setpoint_SO2, setpoint_HCl, initial_SO2,
initial_HCL, initial_H2O, initial_O2):
    # PID controllers
    Kp_SO2, Ki_SO2, Kd_SO2 = 0.01, 15, 0.05
    Kp_HCl, Ki_HCl, Kd_HCl = 0.015, 0.15, 0.01
    pid_SO2 = PIDController(Kp_SO2, Ki_SO2, Kd_SO2, setpoint_SO2)
    pid_HCl = PIDController(Kp_HCl, Ki_HCl, Kd_HCl, setpoint_HCl)

    # Plotted variables
    downstream_so2_values = [initial_SO2]
    downstream_hcl_values = [initial_HCL]
    upstream_so2_values = [initial_SO2]
    upstream_hcl_values = [initial_HCL]
    lime_doses = []

    # Other variables
    downstream_SO2 = initial_SO2 # [mg/Nm3]
    downstream_HCL = initial_HCL # [mg/Nm3]
    h2o_level = initial_H2O
    o2_level = initial_O2
    cf_SO2 = 0
    cf_HCl = 0
    efficiency_so2 = 0.04
    efficiency_hcl = 0.28
    reset_integral_SO2 = 0
    reset_integral_HCl = 0
```

```
    integral_so2 = 0
    integral_hcl = 0
    delay_reading = 6
    delay_estimate = 6
    downstream_SO2_rd = deque([0] * delay_reading, maxlen=delay_reading)
    downstream_HCL_rd = deque([0] * delay_reading, maxlen=delay_reading)
    downstream_SO2_d = deque([0] * delay_reading, maxlen=delay_estimate)
    downstream_HCL_d = deque([0] * delay_reading, maxlen=delay_estimate)
    correction_SO2 = 0
    correction_HCL = 0

    # Simulation:
    for step in range(steps):
        # Get values from dataset
        upstream_HCL = hcl_values_us[step]
        upstream_SO2 = so2_values_us[step]
        fg_flow = fg_flow_values[step]
        h2o_level = h2o_values[step]
        o2_level = o2_values[step]

        # Calculate correcting factors for upstream emissions
        cf_h2o = (100-h2o_level)/100
        cf_o2 = (20.95-o2_level)/(20.95-11)

        # Perform gain scheduling
        Kp_SO2, Ki_SO2, Kd_SO2, Kp_HCl, Ki_HCl, Kd_HCl =
gain_schedule(upstream_SO2, upstream_HCL)
        pid_SO2.update_gains(Kp_SO2, Ki_SO2, Kd_SO2)
        pid_HCl.update_gains(Kp_HCl, Ki_HCl, Kd_HCl)


        # Reset integrals if needed
        integral_so2 = pid_SO2.get_integral()
        integral_hcl = pid_HCl.get_integral()
        if (integral_so2 < 0 and integral_hcl > 0): reset_integral_SO2 = 1
        else: reset_integral_SO2 = 0
        if (integral_hcl < 0 and integral_so2 > 0): reset_integral_HCl = 1
        else: reset_integral_HCl = 0

        # Transform upstream values to kg/h
        upstream_HCL_kg_h = upstream_HCL* fg_flow/10**6
        upstream_SO2_kg_h = upstream_SO2* fg_flow/10**6

        # Calculate correcting factors
        cf_SO2 = pid_SO2.calculate(downstream_SO2+correction_SO2,
reset_integral_SO2)
        cf_HCl = pid_HCl.calculate(downstream_HCL+correction_HCL,
reset_integral_HCl)

        # Calculate lime dosing [kg/h]
        lime_dosing = lime_dosing_control(upstream_SO2, upstream_HCL,
setpoint_SO2, setpoint_HCl, fg_flow, cf_SO2, cf_HCl)
        lime_doses.append(lime_dosing)
```

```
        # Calculate neutralised pollutants [kg/h]
        neutralized_SO2 = max((lime_dosing * 64) / 74 * efficiency_so2,0)
        neutralized_HCL = max((lime_dosing * (36.45 * 2)) / 74 *
efficiency_hcl,0)

        # Calculate downstream values [kg/h]
        downstream_SO2_kg_h = max(0, upstream_SO2_kg_h - neutralized_SO2)
        downstream_HCL_kg_h = max(0, upstream_HCL_kg_h - neutralized_HCL)

        # Transform downstream values to mg/Nm3
        downstream_SO2 = downstream_SO2_kg_h / (fg_flow/10**6)
        downstream_HCL = downstream_HCL_kg_h / (fg_flow/10**6)

        #Apply correcting factors
        downstream_HCL *= cf_h2o*cf_o2
        downstream_SO2 *= cf_h2o*cf_o2

        # Append values for plotting
        downstream_SO2_d.append(downstream_SO2)
        downstream_HCL_d.append(downstream_HCL)
        downstream_SO2_rd.append(downstream_SO2)
        downstream_HCL_rd.append(downstream_HCL)
        correction_SO2 = downstream_SO2_rd[0] - downstream_SO2_d[0]
        correction_HCL = downstream_HCL_rd[0] - downstream_HCL_d[0]

        downstream_so2_values.append(downstream_SO2_rd[0])
        downstream_hcl_values.append(downstream_HCL_rd[0])
        upstream_so2_values.append(upstream_SO2)
        upstream_hcl_values.append(upstream_HCL)

    return downstream_so2_values, downstream_hcl_values, lime_doses,
upstream_so2_values, upstream_hcl_values

for file in file_list:
    print(f"Processing {file}...")

    # Load the CSV file
    data = pd.read_csv(file, low_memory=False)

    # List of column names to check
    columns = ['SO2 Upstream', 'HCl Upstream', 'AirFlow', 'Lime Injected', 'H2O',
'O2 dry', 'SO2 Downstream corrected', 'HCl Downstream corrected']

    # Select the range
    data_subset = data[columns][3:8640]

    # Clean Data
    clean_data = data_subset.replace(['#REF!', 'NaN', '', None],
float('nan')).dropna()

    # Find rows with NaN values
    invalid_rows = clean_data.isna().any(axis=1)
```

```
    # Keep only valid rows
    clean_data = clean_data[~invalid_rows]

    # Convert each cleaned column to a NumPy array
    so2_values_us = clean_data['SO2 Upstream'].astype(float).values
    hcl_values_us = clean_data['HCl Upstream'].astype(float).values
    fg_flow_values = clean_data['AirFlow'].astype(float).values
    actual_lime_dosing = clean_data['Lime Injected'].astype(float).values
    h2o_values = clean_data['H2O'].astype(float).values
    o2_values = clean_data['O2 dry'].astype(float).values
    so2_values_ds = clean_data['SO2 Downstream corrected'].astype(float).values
    hcl_values_ds = clean_data['HCl Downstream corrected'].astype(float).values
    mean_so2_ds = np.mean(so2_values_ds)
    v_actual_avg_so2.append(mean_so2_ds)
    actual_avg_so2 += mean_so2_ds
    mean_hcl_ds = np.mean(hcl_values_ds)
    v_actual_avg_hcl.append(mean_hcl_ds)
    actual_avg_hcl += mean_hcl_ds
    mean_lime_dosing = np.mean(actual_lime_dosing)
    actual_avg_lime += mean_lime_dosing

    # Example usage
    setpoint_SO2 = 40/0.8  # [mg/Nm3]
    setpoint_HCl = 8/0.6  # [mg/Nm3]
    initial_SO2 = so2_values_us[0]  # [mg/Nm3]
    initial_HCL = hcl_values_us[0]  # [mg/Nm3]
    initial_H2O = h2o_values[0] # [%]
    initial_O2 = o2_values[0]
    steps = 8600
    downstream_so2_values, downstream_hcl_values, lime_doses,
upstream_so2_values, upstream_hcl_values = simulate_pid_controller(steps,
setpoint_SO2, setpoint_HCl, initial_SO2, initial_HCL, initial_H2O, initial_O2)
    avg_so2 += np.mean(downstream_so2_values)
    avg_hcl += np.mean(downstream_hcl_values)
    avg_lime += np.mean(lime_doses)
    v_avg_so2.append(np.mean(downstream_so2_values))
    v_avg_hcl.append(np.mean(downstream_hcl_values))


    # Show mean values
    print(f'Mean SO2 emissions: {np.mean(downstream_so2_values):.2f} mg/Nm3 vs
Actual: {mean_so2_ds:.2f} mg/Nm3')
    print(f'Mean HCl emissions: {np.mean(downstream_hcl_values):.2f} mg/Nm3 vs
Actual: {mean_hcl_ds:.2f} mg/m3')
    print(f'Mean Lime Dosage: {np.mean(lime_doses):.2f} kg/h vs Actual:
{mean_lime_dosing:.2f} kg/h\n')

# Plot results
print('\nSummary')
print(f'Mean SO2 emissions: {avg_so2/12:.2f} mg/Nm3 vs Actual:
{actual_avg_so2/12:.2f} mg/Nm3')
```

```
print(f'Mean HCl emissions: {avg_hcl/12:.2f} mg/Nm3 vs Actual:
{actual_avg_hcl/12:.2f} mg/m3')
print(f'Mean Lime Dosage: {avg_lime/12:.2f} kg/h vs Actual:
{actual_avg_lime/12:.2f} kg/h\n')

plt.figure(figsize=(10, 6))

# Upstream values [mg/Nm3]
plt.subplot(3, 1, 1)
plt.plot(upstream_so2_values, label="Upstream SO2", color='r')
plt.plot(upstream_hcl_values, label="Upstream HCl", color='b')
plt.title("Upstream SO2 & HCl Concentration")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

# Downstream values [mg/Nm3]
plt.subplot(3, 1, 2)
plt.plot(downstream_so2_values, label="SO2 Concentration", color='r')
plt.plot(downstream_hcl_values, label="HCl Concentration", color='b')
plt.axhline(y=setpoint_SO2, color='r', linestyle='--', label="Setpoint SO2")
plt.axhline(y=setpoint_HCl, color='b', linestyle='--', label="Setpoint HCl")
plt.title("Downstream SO2 & HCl Concentration")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

# Lime doses [kg/h]
plt.subplot(3, 1, 3)
plt.plot(lime_doses, label="Simulated Lime Dosing", color='green')
plt.plot(actual_lime_dosing, label="Actual Lime Dosing", color='purple')
plt.title("Lime Dosing Output")
plt.xlabel("Time Step")
plt.ylabel("Lime Dosing [kg/h]")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

days = np.arange(1, len(v_avg_so2) + 1)

plt.figure(figsize=(10, 6))

plt.subplot(2, 1, 1)
plt.plot(days, v_avg_so2, label="Simulation Values", color='r')
for i in range(len(v_avg_so2)):
    v_avg_so2[i] *= 0.8
plt.plot(days, v_avg_so2, label="Sim. Values Corrected", color='r', linestyle='--
')
plt.plot(days, v_actual_avg_so2, label="Real Values", color='b')
```

```
for i in range(len(v_actual_avg_so2)):
    v_actual_avg_so2[i] *= 0.8
plt.plot(days, v_actual_avg_so2, label="Real Values Corrected", color='b',
linestyle='--')
plt.axhline(y=40, color='purple', linestyle='--', label="Setpoint SO2")
plt.title("Average Daily SO2 Emissions")
plt.xlabel("Month")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)
plt.xticks(days)

plt.subplot(2, 1, 2)
plt.plot(days, v_avg_hcl, label="Simulation Values", color='r')
for i in range(len(v_avg_hcl)):
    v_avg_hcl[i] *= 0.6
plt.plot(days, v_avg_hcl, label="Sim. Values Corrected", color='r', linestyle='--
')
plt.plot(days, v_actual_avg_hcl, label="Real Values", color='b')
for i in range(len(v_actual_avg_hcl)):
    v_actual_avg_hcl[i] *= 0.6
plt.plot(days, v_actual_avg_hcl, label="Real Values Corrected", color='b',
linestyle='--')
plt.axhline(y=8, color='purple', linestyle='--', label="Setpoint HCl")
plt.title("Average Daily HCl Emissions")
plt.xlabel("Month")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)
plt.xticks(days)

plt.tight_layout()
plt.show()
```

APPENDIX VI –

DAILY_CONTROLLERVISUALISER.PY

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import glob
from natsort import natsorted
from collections import deque

# Constants
MAX_SCREW = 1380 * 1  # [kg/h]
MIN_SCREW = 1380 * 0.09  # [kg/h]

# Get list of available datasets
file_list = glob.glob("limeDosingControl_*.csv")
file_list = natsorted(file_list)
print("Available datasets:")
for i, file in enumerate(file_list):
    print(f"{i+1}: {file}")

# Allow user to choose a dataset
dataset_index = int(input("Select dataset number: ")) - 1
selected_file = file_list[dataset_index]

# Load the CSV file
data = pd.read_csv(selected_file, low_memory=False)

# List of column names to check
columns = ['SO2 Upstream', 'HCl Upstream', 'AirFlow', 'Lime Injected', 'H2O', 'O2
dry', 'SO2 Downstream corrected', 'HCl Downstream corrected']

# Select the range
data_subset = data[columns][3:8640]

# Clean Data
clean_data = data_subset.replace(['#REF!', 'NaN', '', None],
float('nan')).dropna()

# Find rows with NaN values
invalid_rows = clean_data.isna().any(axis=1)

# Keep only valid rows
clean_data = clean_data[~invalid_rows]

# Convert each cleaned column to a NumPy array
```

```
so2_values_us = clean_data['SO2 Upstream'].astype(float).values
hcl_values_us = clean_data['HCl Upstream'].astype(float).values
fg_flow_values = clean_data['AirFlow'].astype(float).values
actual_lime_dosing = clean_data['Lime Injected'].astype(float).values
h2o_values = clean_data['H2O'].astype(float).values
o2_values = clean_data['O2 dry'].astype(float).values
so2_values_ds = clean_data['SO2 Downstream corrected'].astype(float).values
hcl_values_ds = clean_data['HCl Downstream corrected'].astype(float).values
mean_so2_ds = np.mean(so2_values_ds)
mean_hcl_ds = np.mean(hcl_values_ds)

class PIDController:
    def __init__(self, Kp, Ki, Kd, setpoint):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.setpoint = setpoint
        self.prev_error = 0
        self.integral = 0

    def update_gains(self, Kp, Ki, Kd):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd

    def calculate(self, current_value, reset_integral):
        error = (current_value - self.setpoint)/self.setpoint
        P = self.Kp * error

        if reset_integral == 1:
            self.integral = 0
        else: self.integral += error
        I = self.Ki * self.integral

        D = self.Kd * (error - self.prev_error)
        self.prev_error = error

        output = P + I + D
        return output

    def get_integral(self):
        return self.integral

def gain_schedule(upstream_SO2, upstream_HCL):
    so2_ranges = [
        (120, {
            500: (1, 10, 0.05),
            float('inf'): (0.01, 0, 0.05)
        }),
        (180, {
            500: (1, 6, 0.05),
            float('inf'): (0.01, 10, 0.05)
        }),
```

```
        (250, {
            float('inf'): (0.001, 4, 0.05)
        }),
        (350, {
            float('inf'): (0.001, 2, 0)
        }),
        (float('inf'), {
            float('inf'): (0.001, 2, 0)
        })
    ]

    hcl_ranges = {
        280: (0.01, 0.15, 0.0),
        340: (0.025, 0.1, 0.0),
        500: (0.005, 0.05, 0.0),
        600: (0.005, 0.045, 0.0),
        700: (0.001, 0.035, 0.0),
        float('inf'): (0.001, 0.02, 0)
    }

    def find_gains(value, ranges):
        for limit, gains in ranges:
            if value < limit:
                return gains if isinstance(gains, tuple) else
find_gains(upstream_HCL, gains.items())

    Kp_SO2, Ki_SO2, Kd_SO2 = find_gains(upstream_SO2, so2_ranges)
    Kp_HCl, Ki_HCl, Kd_HCl = find_gains(upstream_HCL, hcl_ranges.items())

    return Kp_SO2, Ki_SO2, Kd_SO2, Kp_HCl, Ki_HCl, Kd_HCl

# Feedforward Lime Dosing Calculation
def feedforward_lime_dosing(fg_flow, upstream_SO2, upstream_HCl, setpoint_SO2,
setpoint_HCl):
    QSO2 = (upstream_SO2 - setpoint_SO2) * fg_flow/10**6  # [kg/h]
    QSO2 = max(0, QSO2)
    QHCL = (upstream_HCl - setpoint_HCl) * fg_flow/10**6 # [kg/h]
    QHCL = max(0, QHCL)

    return QSO2, QHCL

# Lime dosing calculation with Feedforward and PID
def lime_dosing_control(upstream_SO2, upstream_HCl,setpoint_SO2, setpoint_HCl,
fg_flow, cf_SO2, cf_HCL):
    # Lime calculation
    QSO2, QHCL = feedforward_lime_dosing(fg_flow, upstream_SO2, upstream_HCl,
setpoint_SO2, setpoint_HCl)

    lime_needed_SO2 = (QSO2 / 64) * 74 *(1+cf_SO2)
    lime_needed_HCL = (QHCL / (36.45 * 2)) * 74 *(1 + cf_HCL)

    lime_needed = lime_needed_SO2 + lime_needed_HCL
    return np.clip(lime_needed, MIN_SCREW, MAX_SCREW)
```

```
# Simulation function
def simulate_pid_controller(steps, setpoint_SO2, setpoint_HCl, initial_SO2,
initial_HCL, initial_H2O, initial_O2):
    # PID controllers
    Kp_SO2, Ki_SO2, Kd_SO2 = 0.01, 15, 0.05
    Kp_HCl, Ki_HCl, Kd_HCl = 0.015, 0.15, 0.01
    pid_SO2 = PIDController(Kp_SO2, Ki_SO2, Kd_SO2, setpoint_SO2)
    pid_HCl = PIDController(Kp_HCl, Ki_HCl, Kd_HCl, setpoint_HCl)

    # Plotted variables
    downstream_so2_values = [initial_SO2]
    downstream_hcl_values = [initial_HCL]
    upstream_so2_values = [initial_SO2]
    upstream_hcl_values = [initial_HCL]
    lime_doses = []

    # Other variables
    downstream_SO2 = initial_SO2 # [mg/Nm3]
    downstream_HCL = initial_HCL # [mg/Nm3]
    h2o_level = initial_H2O
    o2_level = initial_O2
    cf_SO2 = 0
    cf_HCl = 0
    efficiency_so2 = 0.05
    efficiency_hcl = 0.35
    lime_residue = 0 # [kg/h]
    reset_integral_SO2 = 0
    reset_integral_HCl = 0
    integral_so2 = 0
    integral_hcl = 0
    delay_reading = 6
    delay_estimate = 6
    downstream_SO2_rd = deque([0] * delay_reading, maxlen=delay_reading)
    downstream_HCL_rd = deque([0] * delay_reading, maxlen=delay_reading)
    downstream_SO2_d = deque([0] * delay_reading, maxlen=delay_estimate)
    downstream_HCL_d = deque([0] * delay_reading, maxlen=delay_estimate)
    correction_SO2 = 0
    correction_HCL = 0

    # Simulation:
    for step in range(steps):
        # Get values from dataset
        upstream_HCL = hcl_values_us[step]
        upstream_SO2 = so2_values_us[step]
        fg_flow = fg_flow_values[step]
        h2o_level = h2o_values[step]
        o2_level = o2_values[step]

        # Calculate correcting factors for upstream emissions
        cf_h2o = (100-h2o_level)/100
        cf_o2 = (20.95-o2_level)/(20.95-11)
```

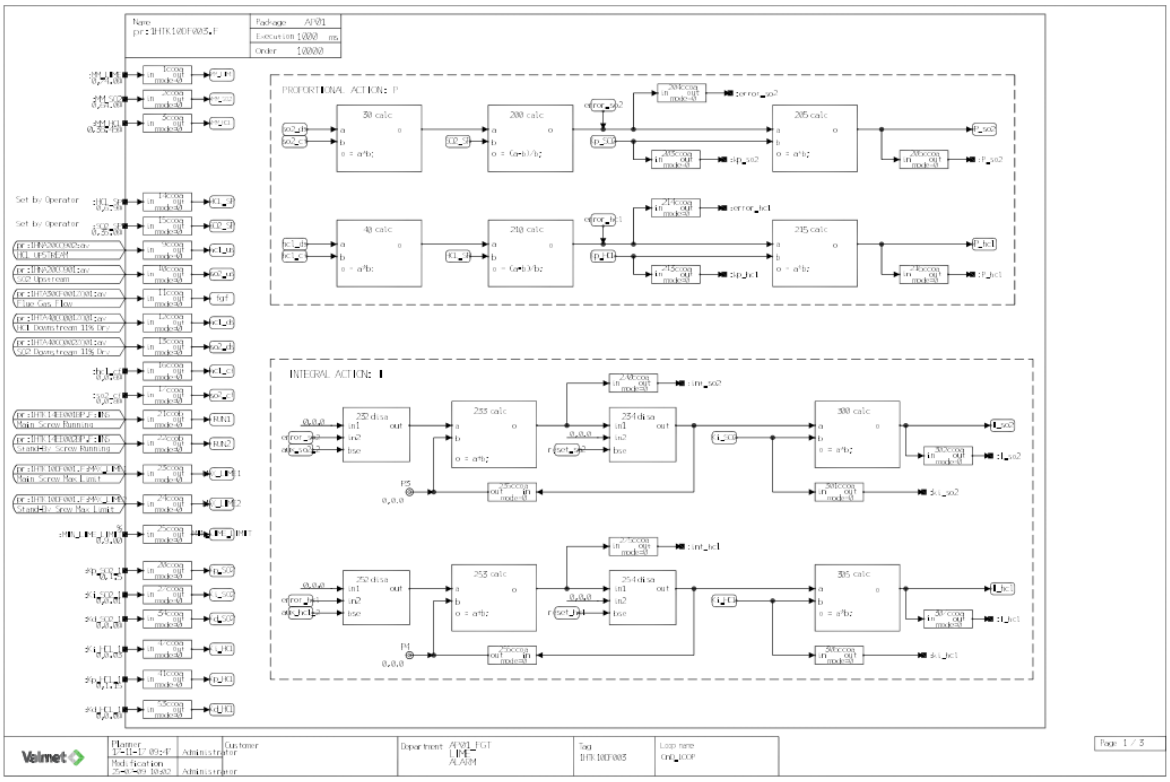
```
        # Perform gain scheduling
        Kp_SO2, Ki_SO2, Kd_SO2, Kp_HCl, Ki_HCl, Kd_HCl =
gain_schedule(upstream_SO2, upstream_HCL)
        pid_SO2.update_gains(Kp_SO2, Ki_SO2, Kd_SO2)
        pid_HCl.update_gains(Kp_HCl, Ki_HCl, Kd_HCl)

        # Reset integrals if needed
        integral_so2 = pid_SO2.get_integral()
        integral_hcl = pid_HCl.get_integral()
        if (integral_so2 < 0 and integral_hcl > 0): reset_integral_SO2 = 1
        else: reset_integral_SO2 = 0
        if (integral_hcl < 0 and integral_so2 > 0): reset_integral_HCl = 1
        else: reset_integral_HCl = 0

        # Transform upstream values to kg/h
        upstream_HCL_kg_h = upstream_HCL* fg_flow/10**6
        upstream_SO2_kg_h = upstream_SO2* fg_flow/10**6

        # Calculate correcting factors
        cf_SO2 = pid_SO2.calculate(downstream_SO2+correction_SO2,
reset_integral_SO2)
        cf_HCl = pid_HCl.calculate(downstream_HCL+correction_HCL,
reset_integral_HCl)

        # Calculate lime dosing [kg/h]
        lime_dosing = lime_dosing_control(upstream_SO2, upstream_HCL,
setpoint_SO2, setpoint_HCl, fg_flow, cf_SO2, cf_HCl)
        lime_doses.append(lime_dosing)

        # Calculate neutralised pollutants [kg/h]
        neutralized_SO2 = max((lime_dosing * 64) / 74 * efficiency_so2,0)
        neutralized_HCL = max((lime_dosing * (36.45 * 2)) / 74 *
efficiency_hcl,0)

        # Calculate downstream values [kg/h]
        downstream_SO2_kg_h = max(0, upstream_SO2_kg_h - neutralized_SO2)
        downstream_HCL_kg_h = max(0, upstream_HCL_kg_h - neutralized_HCL)

        # Transform downstream values to mg/Nm3
        downstream_SO2 = downstream_SO2_kg_h / (fg_flow/10**6)
        downstream_HCL = downstream_HCL_kg_h / (fg_flow/10**6)

        #Apply correcting factors
        downstream_HCL *= cf_h2o*cf_o2
        downstream_SO2 *= cf_h2o*cf_o2

        # Append values for plotting
        downstream_SO2_d.append(downstream_SO2)
        downstream_HCL_d.append(downstream_HCL)
        downstream_SO2_rd.append(downstream_SO2)
        downstream_HCL_rd.append(downstream_HCL)
        correction_SO2 = downstream_SO2_rd[0] - downstream_SO2_d[0]
        correction_HCL = downstream_HCL_rd[0] - downstream_HCL_d[0]
```

```
        downstream_so2_values.append(downstream_SO2_rd[0])
        downstream_hcl_values.append(downstream_HCL_rd[0])
        upstream_so2_values.append(upstream_SO2)
        upstream_hcl_values.append(upstream_HCL)

    return downstream_so2_values, downstream_hcl_values, lime_doses,
upstream_so2_values, upstream_hcl_values

# Example usage
setpoint_SO2 = 39.5  # [mg/Nm3]
setpoint_HCl = 7.8  # [mg/Nm3]
initial_SO2 = so2_values_us[0]  # [mg/Nm3]
initial_HCL = hcl_values_us[0]  # [mg/Nm3]
initial_H2O = h2o_values[0] # [%]
initial_O2 = o2_values[0]
steps = 8600
lim_l = 0
lim_h = 8600

downstream_so2_values, downstream_hcl_values, lime_doses, upstream_so2_values,
upstream_hcl_values = simulate_pid_controller(steps, setpoint_SO2, setpoint_HCl,
initial_SO2, initial_HCL, initial_H2O, initial_O2)

# Show mean values
print(f'Mean SO2 emissions: {np.mean(downstream_so2_values):.2f} mg/Nm3 vs
Actual: {mean_so2_ds:.2f} mg/Nm3')
print(f'Mean HCl emissions: {np.mean(downstream_hcl_values):.2f} mg/Nm3 vs
Actual: {mean_hcl_ds:.2f} mg/m3')
print(f'Mean Lime Dosage: {np.mean(lime_doses):.2f} kg/h vs Actual:
{np.mean(actual_lime_dosing[0:8000]):.2f} kg/h')

# Plot results
upstream_so2_values = upstream_so2_values[lim_l:lim_h]
upstream_hcl_values = upstream_hcl_values[lim_l:lim_h]
downstream_so2_values = downstream_so2_values[lim_l:lim_h]
downstream_hcl_values = downstream_hcl_values[lim_l:lim_h]
lime_doses = lime_doses[lim_l:lim_h]
actual_lime_dosing = actual_lime_dosing[lim_l:lim_h]

plt.figure(figsize=(10, 6))

# Upstream values [mg/Nm3]
plt.subplot(3, 1, 1)
plt.plot(upstream_so2_values, label="Upstream SO2", color='r')
plt.plot(upstream_hcl_values, label="Upstream HCl", color='b')
plt.title("Upstream SO2 & HCl Concentration")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

# Downstream values [mg/Nm3]
```

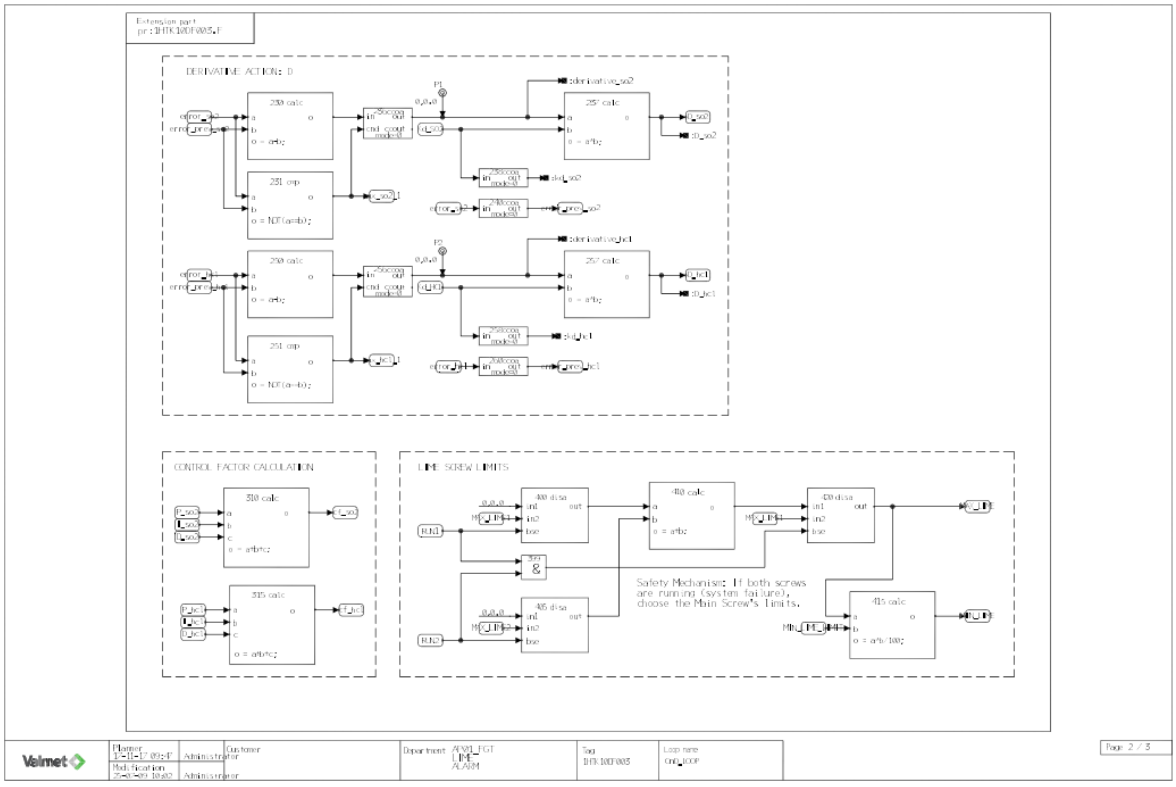
```
plt.subplot(3, 1, 2)
plt.plot(downstream_so2_values, label="SO2 Concentration", color='r')
plt.plot(downstream_hcl_values, label="HCl Concentration", color='b')
plt.axhline(y=setpoint_SO2, color='r', linestyle='--', label="Setpoint SO2")
plt.axhline(y=setpoint_HCl, color='b', linestyle='--', label="Setpoint HCl")
plt.title("Downstream SO2 & HCl Concentration")
plt.xlabel("Time Step")
plt.ylabel("Concentration [mg/Nm3]")
plt.legend()
plt.grid(True)

# Lime doses [kg/h]
plt.subplot(3, 1, 3)
plt.plot(lime_doses, label="Simulated Lime Dosing", color='green')
plt.plot(actual_lime_dosing, label="Actual Lime Dosing", color='purple')
plt.title("Lime Dosing Output")
plt.xlabel("Time Step")
plt.ylabel("Lime Dosing [kg/h]")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

APPENDIX VII – DCS CONTROL LOOP

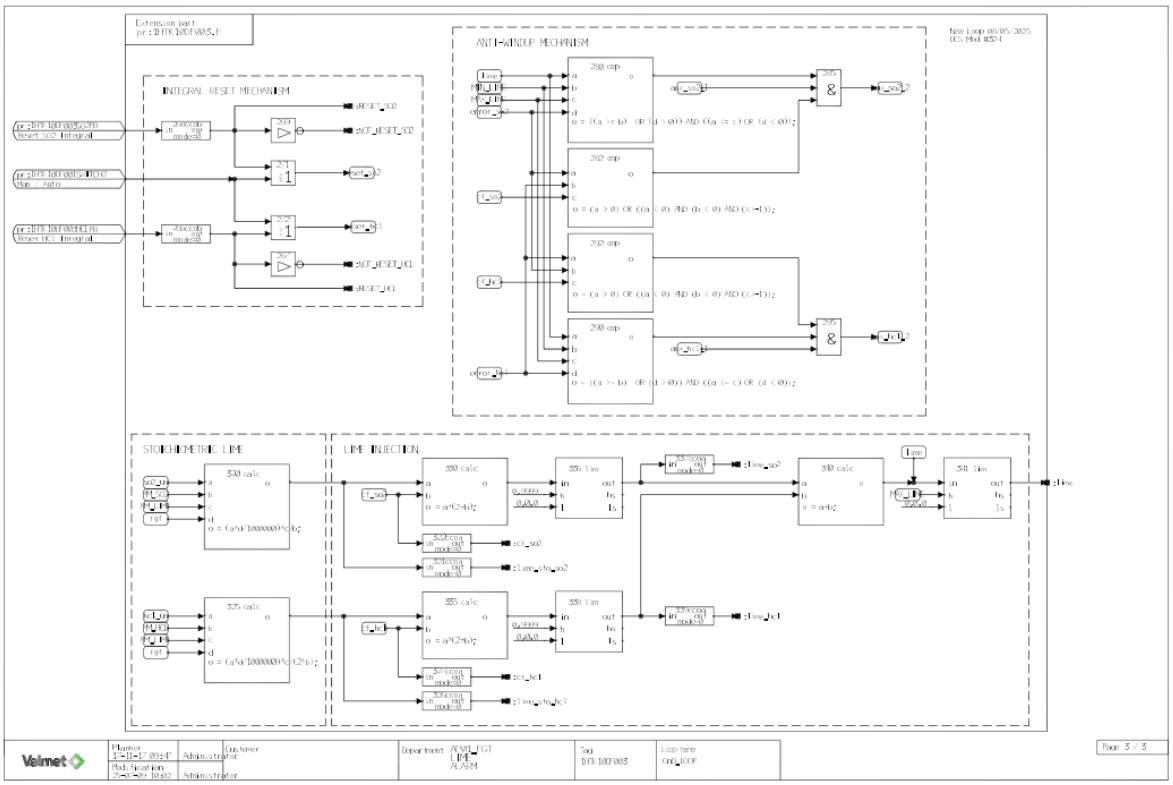
Page 1/3



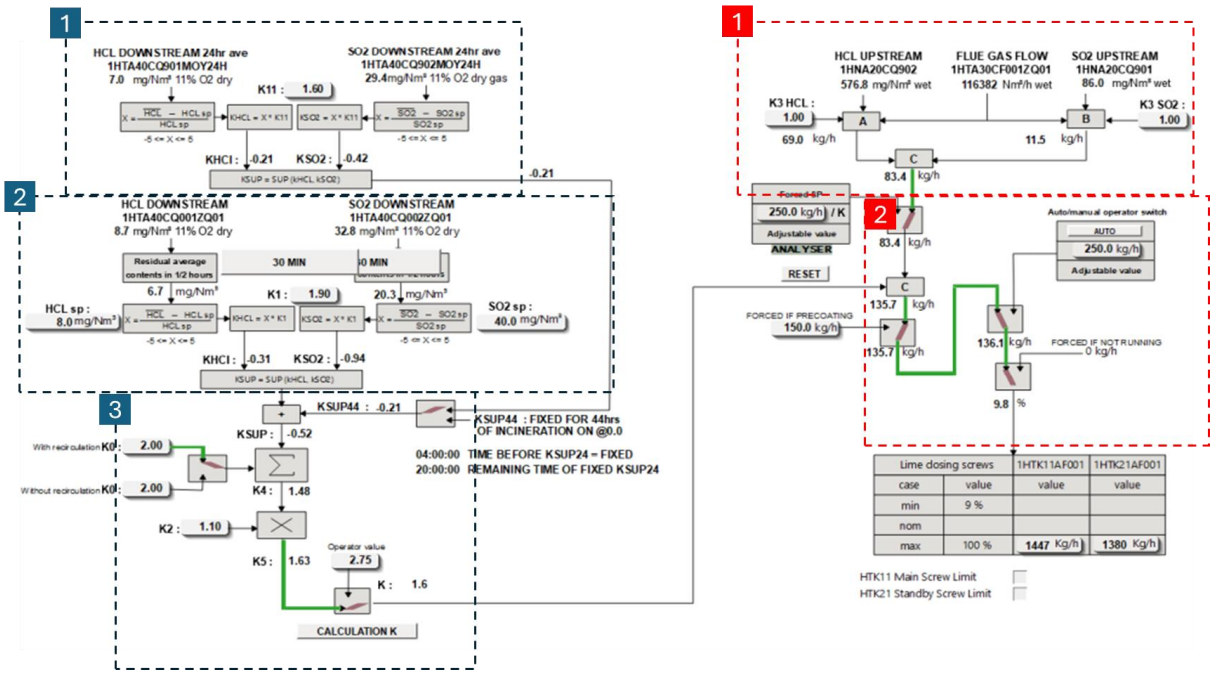
Page 2/3


DERIVATIVE ACTION: D
CONTROL FACTOR CALCULATION
LIME SCREW LIMITS
Safety Mechanism: If both screws are running (system failure), choose the Main Screw's Limits.

Page 3/3


INTEGRAL RESET MECHANISM
ANTI-WINDUP MECHANISM
STOICHIOMETRIC LIME
LIME INJECTION
Valmet

APPENDIX VIII – ORIGINAL INTERFACE



The legacy interface is structured into 2 primary sections, each corresponding to a critical function withing the lime dosing control logic:

1. **Control Loop (Highlighted in blue)**:

This section outlines the logic to obtain the correction factor. It has 3 differentiated blocks:

1.1. 24-Hour Set Point Deviation (K24):

Calculates the maximum deviation of the 24-hour average from the SO_2 and HCl set points.

1.2. 30-Minute Set Point Deviation (K30):

Measures the maximum deviation of the 30-minute average from the respective emission set points.

1.3. Operator Factor (K0-2):

Consists of two **manual adjustment** coefficients (K0 and K2), introduced by the operator based on real-time trend analysis of downstream emissions. In practice, K0 is rarely modified, making the system functionally equivalent to a **single-input control loop**.

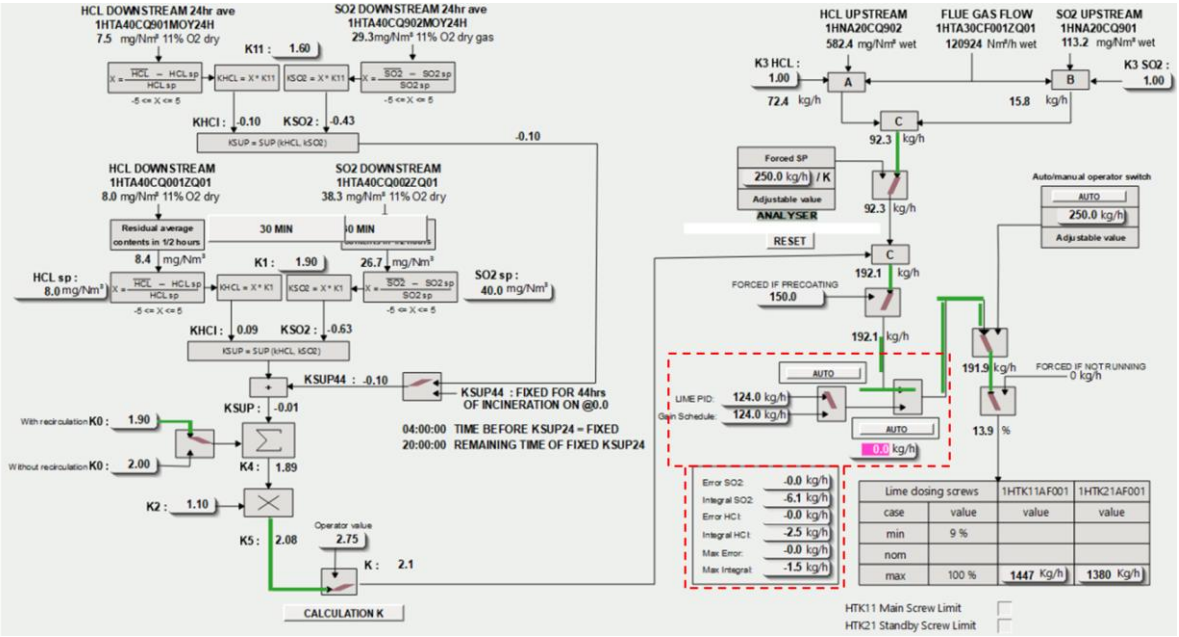
The final correction used to modify lime injection is determined as the product of the K2 factor and the sum of K2, and the K24 and K30 deviation terms (as explained in Section 3.).

2. Stoichiometric and Dosing Lime Calculation (Highlighted in red):

This section computes the required **lime injection** by multiplying the **stoichiometric lime** – calculated based on pollutant concentrations and flue gas flow – with the **correction factor** from the control loop.

Additionally, this area includes a **manual/automatic mode selector switch**, allowing the operator to override the system if necessary.

APPENDIX IX – MODIFIED INTERFACE

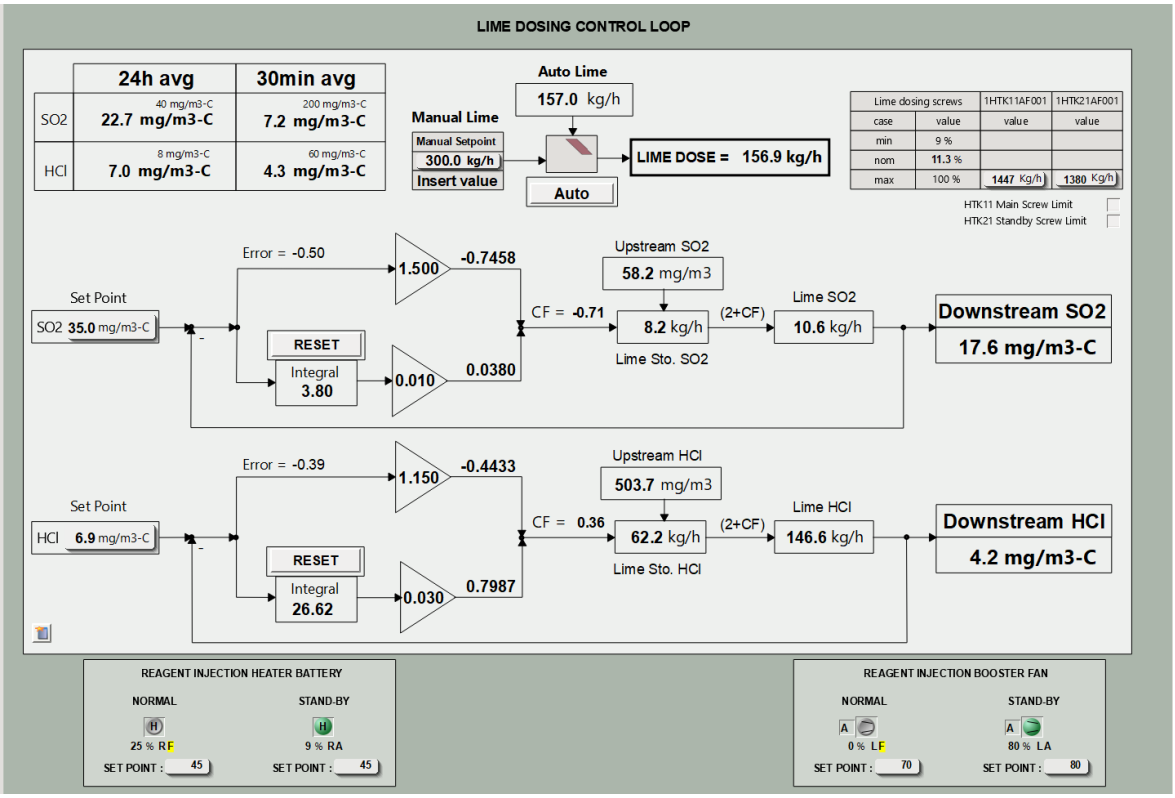


This interface update was designed to seamlessly integrate the new controller into the existing DCS environment without altering the legacy system's logic. The main modifications included:

- **Controller Selection Switch**: A toggle was added to allow operators to manually select between the original and the new controller. This ensured safe fallback capability during the testing phase.
- **New Controller Status Display**: A dedicated section was added to show whether the new controller was active, and to display key indicators such as lime dosing rate, setpoints, and control factor in real-time.
- **Diagnostics Area**: Real-time visualization of internal controller variables.

The updated interface aimed to provide **transparency and operator awareness** during the initial testing and validation phase of the new control system.

APPENDIX X – FINAL INTERFACE



The final interface design focuses on **clarity, control reliability**, and **safe operator interaction**, offering a comprehensive visualisation of the lime dosing control loop. It includes the following key features:

- **Streamlined Layout**:

The interface presents process variables, controller outputs, setpoints, and calculated values in a **logical and structured format**. Elements are grouped by pollutant (SO_2 and HCl), allowing quick assessment of the system's performance. Key indicators such as upstream and downstream concentrations, stoichiometric lime requirements, and final lime dosing rates are clearly displayed.

- **Controlled Operator Interaction**:

Operator input is intentionally limited to **adjusting the emission setpoints** for SO_2 and HCl and **manually resetting the integral term** when needed. The **RESET** buttons are accessible and allow the operator to clear the accumulated error manually, typically after periods of

manual mode operation or extended downtimes. This ensures flexibility while preserving system stability.

- **Live Control Visualisation**:

The interface dynamically reflects the current state of the controller, displaying:

- **Error values** relative to the setpoint.
- **Proportional and integral components** in real time. Derivative components were omitted due to $\boldsymbol{K_D = 0}$.
- **Correction factor (CF)** computation and its impact on the lime dosing output.
- A comparison between **manual and automatic lime setpoints**, clearly showing the current dosing mode.

- **Integrated Safety Indicators**:

Real-time readings of 24-hour and 30-minute emission averages are provided for both SO_2 and HCl, alongside compliance thresholds. Lime dosing limits, screw speeds, and alarms are visually integrated to support safe operation.

This interface not only enables precise control and diagnostics but also reinforces the project's goals of **automation**, **efficiency**, and **operator-friendly design**. Its clarity and functionality help ensure that the system remains robust under varying conditions while limiting the possibility of human error.

APPENDIX XI – HAZOP STUDY

Guide Words	Deviation	Possible Cause	Consequences	Protection Means	Action Required	RAMS
Pollutants	Overemission	Wrong tunning of the controller	Breach	A selector switch has been included in the DCS' interface to allow the operator to toggle between the previous and the new controller.	Operators will perform hourly checks to ensure the pollutants are under compliant limits.	-
				During the tunning stage, the new controller will only be running under the developers' (Carlos & Darío) supervision.	-	-
				HCl Alarm	Operators will adress the HH Level 1HTA40CQ001ZQ24 Alarm switching to manual mode to bring back the emissions under safe limits and eventually switch back to the previous controller.	-
				SO2 Alarm	Operators will adress the HH Level 1HTA40CQ002ZQ24 Alarm switching to manual mode to bring back the emissions under safe limits and eventually switch back to the previous controller.	-
Lime	Overdosing	Wrong tunning of the controller	Baghouse Block	Pressure sensors in the baghouse will turn on the Differential Pressure Alarm (1HTE10CP001XQ02) if it surpasses the HH Level.	Operators will need to open up a door in the baghouse and tamp out the excessive lime.	LN-02-01-OP-OPSBH-004-01-MS LN-02-01-OP-OPSBH-001-01-MS
Maintenance	-	Controller tunning	Overdosing / Underdosing	The new controller has been programmed into the DCS using its local language, FB Cad, making it understandable to anybody with DCS experience.	-	-
				There's a detailed manual available on the controller's functioning and tuning.	An authorised person with DCS knowledge should modify the gains from the loop described in the 1HTK10DF003 file.	Lime_Dosing_System_Operations_Manual.pdf
	Integral Windup	Switching off the Controller over an extended period (such an outage)	Large Accumulation of Errors	Integral Reset button.	Press the button to bring the integral back to 0.	Lime_Dosing_System_Operations_Manual.pdf
				Automatic Integral Reset.	If the controller is turned into manual mode, the integral will be automatically reset.	Lime_Dosing_System_Operations_Manual.pdf
Operations	-	Operators trained to understand the lime dosing control in the DCS	Overdosing / underdosing / breach	-	Operators to read the training document.	Lime_Dosing_System_Operations_Manual.pdf

Table 4 HAZOP Study Summary Table.

APPENDIX XII – FINAL REPORT

This appendix provides supplementary information relevant to the Final Report, complementing the final conclusions presented in Section 5.

*NM = Near Miss

Final Report														
Day	Avg. Lime Dose	Lime Dose/STO	Lime STO	SO2 SP Distance	SO2 Avg	SO2 STDV	SO2 Cv	HCl SP Distance	HCl Avg	HCl STDV	HCl Cv	Lime/Waste	1/2 Hour NM	24 Hour NM
Unit	kg/h	-	kg/h	-	mg/m3	mg/m3	-	-	mg/m3	mg/m3	-	kg/t	-	-
13/06/2024	208	2.86	72.73	-8.2%	36.7	25.0	68%	22.5%	9.8	2.2	22%	9.15	0	No
13/06/2025	165	2.14	76.78	-17.6%	24.7	15.1	61%	-3.9%	5.8	1.3	22%	7.56	0	No
14/06/2024	210	3.12	67.12	12.9%	45.2	27.8	62%	26.9%	10.2	2.5	24%	9.03	0	No
14/06/2025	226	2.52	89.50	-15.8%	25.3	17.5	69%	-2.5%	5.9	1.8	31%	9.31	0	No
15/06/2024	217	2.88	75.33	18.4%	47.3	24.6	52%	10.4%	8.8	1.5	17%	9.60	0	No
15/06/2025	220	2.30	95.62	-23.4%	23.0	16.2	70%	-1.3%	5.9	1.6	27%	9.33	0	No
16/06/2024	232	2.98	77.71	12.6%	45.0	26.8	59%	18.4%	9.5	2.1	22%	10.15	0	No
16/06/2025	206	2.11	98.03	-26.4%	25.7	19.1	74%	-1.1%	6.9	2.0	29%	9.17	0	No
17/06/2024	225	2.95	76.34	3.1%	41.3	26.2	63%	19.1%	9.5	2.3	24%	10.26	0	No
17/06/2025	215	2.17	98.78	-27.1%	25.5	17.9	70%	-0.8%	6.9	1.8	26%	10.36	0	No
18/06/2024	242	2.97	81.30	-7.2%	37.1	27.1	73%	28.9%	10.3	2.7	26%	11.16	0	No
18/06/2025	210	2.10	100.05	-34.8%	22.8	17.3	76%	-0.7%	6.9	1.9	27%	10.37	0	No
19/06/2024	238	3.18	74.78	-9.5%	36.2	25.8	71%	32.2%	10.6	3.0	28%	10.51	0	No
19/06/2025	184	2.28	80.90	-35.6%	22.5	14.2	63%	-0.7%	7.0	1.6	22%	8.42	0	No
30/06/2024	305	3.49	87.22	1.5%	40.6	25.5	63%	46.5%	11.7	3.1	26%	14.28	0	No
30/06/2025	211	2.07	101.72	-38.7%	21.5	13.7	64%	-0.1%	6.9	1.6	24%	9.75	0	No
01/07/2024	270	3.29	82.12	-15.3%	33.9	21.4	63%	33.3%	10.7	2.5	24%	12.89	0	No
01/07/2025	220	2.19	100.28	-40.4%	20.9	14.1	68%	-0.9%	6.8	1.7	25%	10.87	0	No
02/07/2024	252	3.20	78.59	-12.1%	35.1	20.4	58%	33.1%	10.7	2.4	22%	11.60	0	No
02/07/2025	217	2.19	99.13	-40.7%	20.8	14.6	70%	-0.1%	6.9	2.1	30%	10.66	0	No
03/07/2024	256	3.38	75.72	-7.3%	37.1	21.4	58%	37.6%	11.0	2.2	20%	11.73	0	No
03/07/2025	198	2.15	92.03	-40.6%	20.8	14.1	68%	-1.9%	6.8	1.8	27%	9.49	0	No
04/07/2024	235	3.28	71.56	-7.5%	37.0	21.8	59%	32.9%	10.6	2.1	20%	10.85	0	No
04/07/2025	191	2.21	86.35	-35.7%	22.5	16.2	72%	-1.8%	6.8	2.0	29%	8.90	0	No
05/07/2024	213	3.00	70.95	-5.4%	37.9	21.3	56%	25.6%	10.0	1.9	19%	9.68	0	No
05/07/2025	208	2.14	97.16	-24.1%	26.6	15.0	56%	-0.9%	6.8	1.5	22%	10.03	0	No
06/07/2024	243	3.15	77.00	6.2%	42.5	22.8	54%	33.6%	10.7	2.3	21%	11.47	0	No
06/07/2025	215	2.10	102.29	-26.4%	25.8	17.3	67%	-1.0%	6.8	1.8	26%	10.54	0	No

	Day-to-day comparison												
	Avg. Lime Dose	Lime Dose/STO	Lime STO	SO2 SP Distance	SO2 Avg	SO2 STDV	SO2 Cv	HCl SP Distance	HCl Avg	HCl STDV	HCl Cv	Lime/Waste	1/2 Hour NM	24 Hour NM
Average 2024	238.86	3.12	76.32	-1.3%	39.49	24.13	61%	28.6%	10.29	2.34	23%	10.88	0	0
Average 2025	206.00	2.19	94.19	-30.5%	23.45	15.86	68%	-1.3%	6.65	1.74	26%	9.62	0	0
Difference	-14%	-30%	23%	-	-41%	-34%	10%	-	-35%	-26%	15%	-12%	-	-
Avg. 2025 after change of SP	206.76	2.16	96.07	-33.7%	23.21	15.76	68%	-0.9%	6.87	1.79	26%	9.87	0	0
Difference	-13%	-31%	26%	-	-41%	-35%	11%	-	-33%	-23%	15%	-9%	-	-

	Long-Term Assessment										
	Avg. Lime Dose	Lime Dose/STO	Lime STO	SO2 SP Distance	SO2 Avg	SO2 STDV	SO2 Cv	HCl SP Distance	HCl Avg	HCl STDV	HCl Cv
Average 2024	240.16	3.30	72.79	-9.5%	36.2	24.2	66.9%	2.3%	8.2	2.8	34%
Average 2025	206.00	2.19	94.04	-30.5%	23.4	15.9	68%	-1.3%	6.7	1.7	26%
Difference	-14%	-34%	29%	-	-35%	-34%	1%	-	-19%	-38%	-23%
Avg. 2025 after change of SP	206.76	2.16	96.07	-33.7%	23.21	15.76	68.0%	-0.9%	6.87	1.79	26.1%
Difference	-14%	-35%	32%	-	-36%	-35%	2%	-	-16%	-36%	-23%

Table 5 Final Report.

- The previous control system operated based on downstream emission values *without* applying confidence factors. In contrast, the new control system incorporates these factors into its feedback mechanism.

To ensure a fair and objective comparison between both systems, their performance was evaluated based on the magnitude they were actively addressing.

- A **divider line on June 15** in the table above marks a **change in the system's set points**. During the initial testing phase, **stricter set points** were applied to ensure full regulatory compliance while assessing the controller's behaviour. Once sufficient confidence in the system's accuracy was established – evidenced by an **average HCl deviation of -2.6%** and a **maximum deviation of -3.9%** – the set points were adjusted to more realistic values:
    - **HCl set point** increased from **6.0 mg/Nm³** to **7.0 mg/Nm³.**
    - **SO_2 set point** increased from **30.0 mg/Nm³** to **35.0 mg/Nm³.**

This change reflects a strategic move towards **optimising lime consumption**, while still maintaining **emission levels comfortably within compliance thresholds**.

- Besides from the **regulatory compliance analysis** outlined in Section 5, an additional 2-step assessment was conducted to provide a broader vision of the system's reliability and compliance:
    - **30-minute average assessment**: for each 24-hour report, 30-minute average downstream emissions (with confidence factors applied) were obtained using Excel dynamic tables. Near misses were determined comparing these values with the Environmental Agency's fixed limits, and results demonstrate that regulatory compliance was never compromised.
    - **24-hour assessment**: the daily average downstream emissions (with confidence factors applied) were also compared against regulatory limits. In accordance with the high accuracy presented by the new system, the process always complied with all regulatory requirements.
- The **Lime-to-Waste Ratio** analysis was conducted exclusively within the **day-to-day assessment**, as the large volume of data in the one-year-report, combined with the time constraints of the Project, made it unfeasible to extend this analysis to the long-term evaluation.

- A **10-day gap**, spanning from **19th June** to **30th June**, is present due to the testing of an alternative controller, during which data relevant to this Project's scope was not collected.

APPENDIX XIII – ECONOMIC STUDY

Annual Costs:	£400,000
Cost Savings:	10%
Inflation:	2%

Income						
	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
Savings	-	£40,000	£40,800	£41,616	£42,448	£43,297
Total	**£0**	**£40,000**	**£40,800**	**£41,616**	**£42,448**	**£43,297**

Costs						
	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
CAPEX	£3,378	-	-	-	-	-
OPEX	-	-	-	-	-	-
Total	**£3,378**	**£0**	**£0**	**£0**	**£0**	**£0**

Cash Flow						
	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
Cash Flow	-£3,378	£40,000	£40,800	£41,616	£42,448	£43,297
ACCF	-£3,378	£36,623	£77,423	£119,039	£161,487	£204,784
Amortisation	-	£3,378	-	-	-	-
Payback	-£3,378	£36,623	£40,800	£41,616	£42,448	£43,297

ROI =	61
IRR =	1186%
Payback =	1

APPENDIX XIV – PROJECT'S ALLIGNMENT WITH THE SDGS

This Project aligns with several SDGs:

SDG 7 – Affordable and Clean Energy

The project enhances the efficiency of an Energy-from-Waste (EfW) plant, contributing to cleaner energy production by optimizing pollutant control.

SDG 9 – Industry, Innovation, and Infrastructure

The automation of lime dosing introduces a more advanced and efficient industrial control system.

SDG 12 – Responsible Consumption and Production

The project aims to reduce lime consumption by at least 5%, optimizing resource use and minimizing waste.

SDG 13 – Climate Action

By improving the efficiency of pollutant neutralization (SO_2 and HCl), the project helps mitigate environmental impact and supports cleaner air. Moreover, lower lime consumption reduces the carbon footprint associated with its production and transportation.

SDG 14 & 15 – Life Below Water & Life on Land

Keeping SO_2 and HCl emissions under regulatory limits helps prevent acid rain and environmental degradation, benefiting both terrestrial and aquatic ecosystems.