



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola d'Enginyeria de Barcelona Est

MASTER'S FINAL THESIS

**Master's Degree in Chemical Engineering – Smart Chemical
Factories**

**DEVELOPMENT OF MACHINE LEARNING STRATEGIES FOR
FAULT DIAGNOSIS IN VIRTUAL PLANTS (DIGITAL TWINS)**



Report and Annexes

Author:	Kevin Martínez López
Director:	Moisès Graells Sobré
Co-Director:	Gerard Campanyà Gratacòs
Call date:	June 2021

Resumen

En este proyecto, se ha validado la posibilidad de realizar la monitorización de datos y el diagnóstico de errores en línea (mientras se ejecuta la simulación) de una planta química simulada (Digital Twin en Inglés). La simulación se encuentra funcionando en un ordenador remoto, mientras que se accede a los resultados de la monitorización de datos y el diagnóstico de errores por medio del acceso, con un ordenador personal, a la nube, más conocida como 'Cloud' por su término en Inglés.

En primer lugar, se explica la implementación, módulo a módulo, del prototipo modular propuesto y empleado para el intercambio de información desde el 'Digital Twin' hacia la nube (Cloud), lo que permite la monitorización de datos. Para cada módulo, se introducen los programas y herramientas de programación necesarios para crear y/o ejecutar el módulo. Las razones para seleccionar los programas y las herramientas también son expuestas. Además, se introduce la plataforma donde se aloja la nube empleada junto con los diferentes servicios disponibles en la nube, los cuales se han usado para mostrar los resultados de la monitorización de datos.

En segundo lugar, los algoritmos de aprendizaje automático (Machine Learning en Inglés) y de análisis de datos (Data Analysis en Inglés), implementados para el diagnóstico de fallos, se comentan desde los puntos de vista teórico y de implementación. Además, se explica el desarrollo de las herramientas de monitorización para el diagnóstico de fallos, que consiste en la combinación de los anteriores algoritmos con el prototipo modular encargado del intercambio de información.

Finalmente, se documenta una prueba de concepto del prototipo en global, que demuestra que estas tecnologías son factibles y fiables para la monitorización de datos y el diagnóstico de fallos. Adicionalmente, se incluyen unas pautas a seguir para mejorar el prototipo.

Resum

En aquest projecte, s'ha validat la possibilitat de realitzar la monitorització de dades i el diagnòstic d'errors en línia (mentre s'executa la simulació) d'una planta química simulada (Digital Twin, en Anglès). La simulació es troba funcionant a un ordinador remot, mentre que s'accedeixen als resultats de la monitorització de dades i el diagnòstic d'errors per mitjà de l'accés, amb un ordinador personal, al núvol, més conegut com a 'Cloud' pel seu terme en Anglès.

En primer lloc, s'explica la implementació, mòdul a mòdul, del prototipus modular proposat i emprat per a l'intercanvi d'informació des del 'Digital Twin' cap al núvol, el qual permet la monitorització de dades. Per a cada mòdul, s'introdueixen els programes o eines de programació necessaris per a la creació i/o execució. Les raons considerades alhora d'escollir aquests programes o eines de programació també s'exposen. A més a més, s'introdueix la plataforma on s'allotja el núvol junt amb els diferents servicis que ofereix el núvol, els quals han sigut utilitzats per a mostrar els resultats de la monitorització de dades.

En segon lloc, els algorismes d'aprenentatge automàtic (Machine Learning en Anglès) i d'anàlisi de dades (Data Analysis en Anglès), que han sigut implementats pel diagnòstic d'errors en la planta simulada, són comentats des dels punts de vista teòric i d'implementació. També, s'explica el desenvolupament de les eines de monitorització per a la diagnosi d'errors, les quals són el resultat de combinar els algorismes anteriors amb el prototipus modular encarregat de l'intercanvi d'informació.

Finalment, es documenta una prova de concepte del prototipus global que permet demostrar que aquestes tecnologies son factibles i fiables per a la realització de la monitorització de dades i el diagnòstic d'errors. Addicionalment, s'inclouen pautes a seguir per a millorar el prototipus.

Abstract

In this project, the possibility of performing the on-line data monitoring and fault diagnosis over a simulated chemical plant (Digital Twin) has been validated, which is running on a remote computer, by accessing the Cloud with a personal computer.

Firstly, the implementation is explained, module by module, of the modular prototype proposed and employed for the exchange of information from the Digital Twin to the Cloud, which enables the data monitoring. For each of the modules, the programs or programming tools required for its creation and/or execution are introduced. The reasons for its selection are also exposed when explaining each of the modules. Moreover, the Cloud Platform chosen is also introduced together with the different services associated with it that have been used for displaying the results from data monitoring.

In the second place, the Machine Learning and Data Analysis algorithms implemented for the fault diagnosis are commented from the theoretical and the implementation points of view. Furthermore, the development of monitoring tools for fault diagnosis is also explained, which consists of the coupling between the algorithms and the modular prototype for the exchange of information.

Finally, it is documented a proof of concept of the global prototype, which demonstrates the feasibility and reliability of these technologies for performing data monitoring and fault diagnosis. Additionally, guidelines for the further development of the prototype are provided.



Acknowledgments

On the one hand, I would like to offer my special thanks to both my main supervisor and my second supervisor, who have always been receptive to clarify doubts, share their personal opinion and knowledge, to offer guidance relative to the writing of the report and to how adequately explain the ideas and concepts of such a specific field as the one dealt within this project.

On the other hand, I would like to express my very great appreciation to my family and friends for their support during the realization of the herein project.

Glossary

PFD: Process Flow Diagram

P&ID: Piping and Instruments Diagram

SCADA: Supervisory Control and Data Acquisition software or system

DT: Digital Twin

PDT: Process Digital Twin

OPC UA: Open Platform Communications Unified Architecture. It stands for a communications protocol.

GUI: Graphical User Interface

MQTT: Message Queue Telemetry Transport. It stands for a protocol for transporting messages.

IoT: Internet of Things. It englobes all the physical objects that can exchange data with other devices over the internet.

SAS Token: Shared Access Signature Token. If related to the Cloud, it is used so as to grant access to the person receiving the Token.

SQL: Structured Query Language. Programming tool for the retrieval of information contained in databases.

PCA: Principal Component Analysis

MLA: Machine Learning Algorithm

Table of contents

RESUMEN	I
RESUM	II
ABSTRACT	III
ACKNOWLEDGMENTS	V
GLOSSARY	VI
FOREWORD	1
1.1. Origins of the project	1
1.2. Motivation	1
1.3. Previous requirements	1
INTRODUCTION	3
1.4. State of the art	3
1.5. Objectives of the project	3
1.6. Scope of the project	3
PROTOTYPE FOR THE EXCHANGE OF INFORMATION FROM A DIGITAL TWIN OF A CHEMICAL PLANT	5
1.7. Concept of digital twin and tools for its creation	5
1.8. Process simulation tool for implementing the digital twin	6
1.8.1. UniSim as a tool for dynamics simulation	6
1.8.2. Creation and configuration of a simulated case	6
1.8.3. Description of the simulated case	16
1.9. Communications interface programming and developing	18
1.9.1. Python as programming language	19
1.9.2. Python code functions for communications and monitoring	19
1.9.3. Built-in interface (GUI library Tk)	23
1.9.4. Role of the corresponding code as SCADA	24
1.9.5. Sequence Diagram for communications and monitoring in UML standard	25
1.10. Modular communications architecture	26
1.10.1. Modular architecture	26
1.11. Publication of on-line data through the internet	37
1.11.1. Modular architecture	37
1.11.2. Azure IoT Hub	38
1.11.3. Stream Analytics	39
1.11.4. Power BI	41

1.11.5. Configuration of the message route.....	44
1.11.6. Advantages and disadvantages of the Cloud	45
DEVELOPMENT OF MACHINE LEARNING AND DATA ANALYSIS ALGORITHMS	47
1.12. Theoretical background	47
1.12.1. Normalization of the data and pretreatment steps.....	47
1.12.2. PCA Method.....	48
1.12.3. Mahalanobis distance.....	51
1.13. Python code functions for communications combined with machine learning and data analysis.....	52
1.13.1. Implementation of machine learning and data analysis functions.....	52
1.13.2. Built-in interface (GUI library Tk).....	56
1.14. Sequence Diagram for communications, machine learning and data analysis in UML standard.....	57
1.15. Other potential applications of Cloud computing tools	58
1.15.1. Current potential related to the use of the Cloud.....	58
DEVELOPMENT OF MONITORING TOOLS FOR THE DETECTION OF ABNORMAL EVENTS IN A SIMULATED PLANT	59
1.16. Reading of data by the data analysis and machine learning algorithms.....	59
1.17. Preparation of simulation cases for training and testing the machine learning algorithm	60
1.17.1. Simulation case for training the machine learning algorithm.....	60
1.17.2. Simulation case for testing the machine learning algorithm	61
1.18. Optimization of the Data Analysis and the Machine Learning Algorithms	61
1.19. Fault Detection Palette.....	66
VALIDATION OF THE PROTOTYPE AND GUIDELINES FOR FUTURE DEVELOPMENT	68
1.20. Validation of the prototype.....	68
1.20.1. Proof of concept	68
1.20.2. Analysis of the results obtained during the proof of concept.....	69
1.21. Guidelines for future development	72
CONCLUSIONS	74
ECONOMIC ANALYSIS	76
1.22. Electricity costs	76
1.23. Cloud Platform costs	77
1.24. Personnel costs.....	78
1.25. Total costs.....	79
WEBOGRAPHY	81

Foreword

1.1. Origins of the project

In the chemical industrial sector, there is a continuous generation of data, which has been and is traditionally stored locally for the company to keep the information contained within the data under safety. Nonetheless, the latest advances in technologies from the communications and data management fields have opened an opportunity window for changing this aforementioned traditional behavior. These new technologies can be used in order to switch from just storing data to adopting new methodologies aimed to gain valuable knowledge from this data.

1.2. Motivation

This project is borne with the intention of being a proof of concept. It serves so as to demonstrate that the new technologies from the data communications and data management fields, which are currently being studied in the industry and the academia, render meaningful results. Therefore, they could be further explored in future projects.

From a personal point of view, the project has been a challenge during its realization due to the fact that it is required to possess and to combine multidisciplinary knowledge and skills. However, this challenge has been seen as a continuous source for motivation.

1.3. Previous requirements

For the realization of this project, it has been necessary to have basic knowledge and skills regarding:

- The use of commercial software intended for process simulation related to the chemical industry.
- How to code in programming languages. In this specific case, how to code in Python language.
- Data Analysis and Machine Learning.

Introduction

1.4. State of the art

The topic explored within this project is nurtured from the latest advances on technologies that are popping up in the market and from technology that was already established several years ago. On the one hand, the latest advances in technologies correspond to data analysis, and machine learning tools, and communication networks. On the other hand, the established technology is related to the digital twin concept in the chemical industry.

1.5. Objectives of the project

This project has been performed through the achievement of milestones, each of which is required to be completed prior to continuing with the subsequent milestone. At the same time, each milestone is composed of several objectives. Bearing that in mind, four milestones constitute the project as described below:

- **To develop a prototype for exchanging data from a digital twin of a chemical plant**
 - To select a process simulation tool for implementing the digital twin
 - To develop and program the communications interface
 - To use standard communication protocols and a modular architecture
 - To publish the on-line data through the Internet
- **To develop tools for producing data-based models of the process from this data**
 - To use Machine Learning and Data Analysis algorithms
 - To explore the use of Cloud computing tools
- **To develop monitoring tools for the detection of abnormal events in the simulated plant**
 - To prepare simulation cases for training and testing the machine learning algorithm
 - To demonstrate and assess the use and efficiency of the remote monitoring system
- **To validate the prototype and provide guidelines for future development**
 - To make a proof of concept of the remote monitoring system developed
 - To provide guidelines aimed to the future development of the remote monitoring system

1.6. Scope of the project

A digital twin is to be created with the use of a commercial simulator devoted to process simulation in the chemical industry. The aforementioned digital twin is to be connected to the Cloud, where technologies of data monitoring and fault detection are to be used as a proof of concept.



Prototype for the Exchange of information from a digital twin of a chemical plant

1.7. Concept of digital twin and tools for its creation

In the chemical industry, a digital twin can be defined as a digital copy of a process plant or an individual process, which serves for testing operation strategies and for the training of operators. The concept can be easily understood when its origins are known: flight simulators. The first digital twins were created for the training of airline pilots (1).

The first technologies for simulating a digital twin of a chemical plant were developed back in the 60s when Fortran was being used as a programming tool. There are three alternatives available nowadays for creating a digital twin:

1. Programming languages
2. Generalist simulators
3. Commercial simulators

The use of programming languages for the simulation of a digital twin requires to build-up everything from scratch, to program everything on your own. It is possible but unfeasible for the objectives of this project.

Generalist simulators can be seen as programming environments more sophisticated. Their user interface is high-level, which enables a user-friendly interaction. It is possible the modeling of process units through equation-oriented language or adding control schemes for these process units with building blocks. For instance, Matlab and Simulink, or Modelica are clear examples of these types of simulators.

However, generalist simulators have as drawback the lack of data. They do not include a database with thermodynamic and component packages. Therefore, it is possible to model process units, but the simulation of a realistic behavior of these process units gets hard. In consequence, the modeling of huge and complex processes turns unaffordable.

Commercial simulators are programming environments of very high-level, which include pre-built modules for process units and databases with thermodynamic and component packages. The use of this kind of simulation results attractive in the herein project since it makes it possible to simulate a chemical plant easily having associated realistic behavior.

There are different commercial simulators available, which can be open or closed-source. The most well-spread and known software are Aspen Hysys and Aspen Plus, which are two software products closed-source, property of the company AspenTech. Nevertheless, its costs are quite expensive. There are also open-source options such as DWSIM, but it is not as well-spread or trusted as the aforementioned options. On the other hand, other available options are CHEMCAD, ProSimPlus, gPROMS, Petro-SIM, AVEVA PRO/II, Schumberger's VMG Symmetry or UniSim among others (2).

From the different options available, UniSim has been selected due to being an economically-feasible option and accounting with useful tools for the simulation of errors and abnormal behaviors within the simulated chemical plant.

1.8. Process simulation tool for implementing the digital twin

1.8.1. UniSim as a tool for dynamics simulation

UNISIM® Design Suite R480 has been employed as the process simulation tool so as to recreate a chemical plant. The software enables engineers to create steady-state and dynamic models for plant design, performance monitoring, troubleshooting and operational improvement through the use of different modules (3).

In the herein project, the modules UniSim Design and UniSim Dynamic Option have been used. On the one hand, the former, being a steady-state flow sheeting environment, permits to create a steady-state model by selecting the appropriate thermodynamics properties, feed compositions and conditions, and unit, logic and control operations. On the other hand, the latter converts the steady-state model into a dynamic model with the use of a dynamic's assistant. The dynamic model offers rigorous and high-fidelity results (4).

1.8.2. Creation and configuration of a simulated case

After UNISIM® Design Suite R480 is opened, the option New Case is selected from the different options depicted in Figure 1.

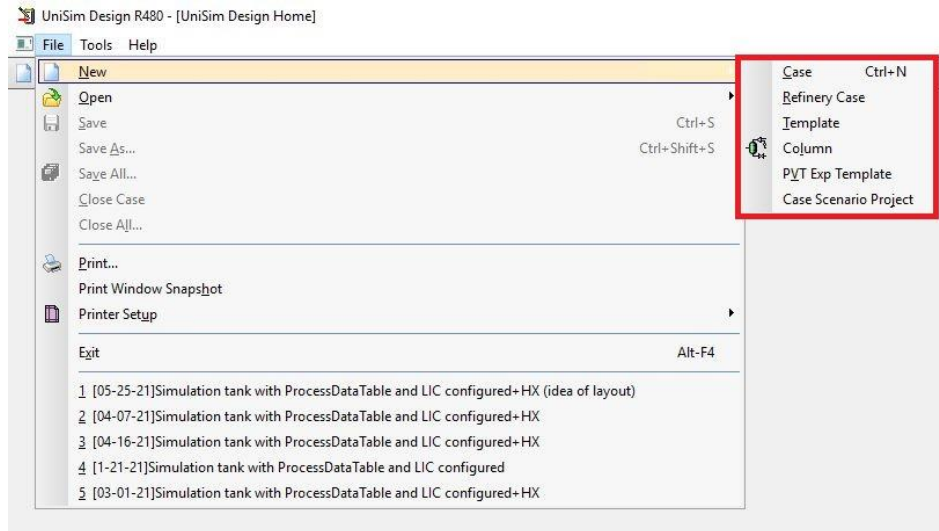


Figure 1. Different available options when creating a new simulation

1.8.2.1. Simulation Basis Manager

A new window named *Simulation Basis Manager* pops-up, which is an environment that permits the user to specify from the internal database of the commercial simulator:

- The different components or chemical substances that will be present in the simulation
- The fluid or thermodynamics package, which includes the physical and chemical properties of the components and the thermodynamics models that will be used in the simulation.

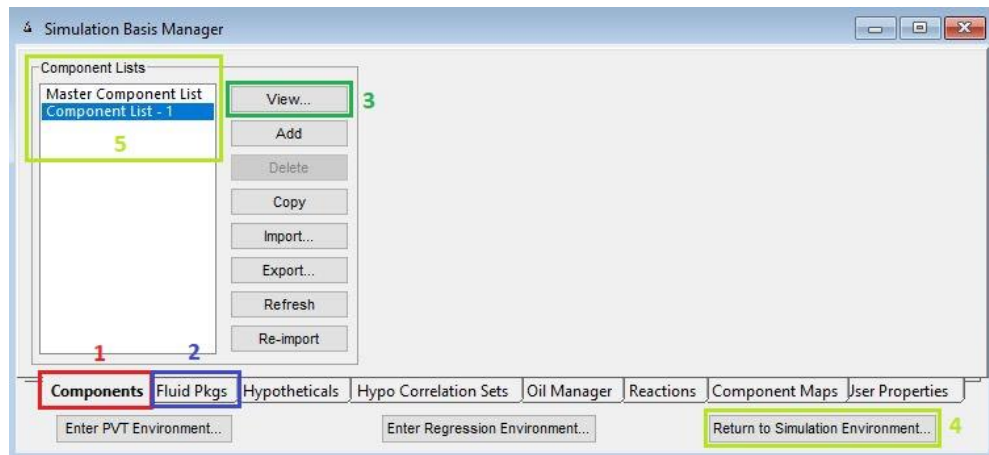


Figure 2. Simulation Basis Manager

According to Figure 2, it is possible to access the component lists when clicking on the tab *Components* (1). The Fluid Package menu can be accessed when clicking on the tab *Fluid Pkgs* (2). Once in 1 or 2, if the tab *View* (3) is clicked, the user accesses to the *Component List View* or the *Fluid Packages View* respectively.

From the windows *Component List View* and *Fluid Packages View* the different components and the fluid package are selected. After those have been established, it is possible to enter in the Simulation Environment in which the user can start to design the Process Plant in steady-state mode. To do so, the tab *Return to Simulation Environment (4)* (if the user has already entered in this environment) or *Enter to Simulation Environment* (if the user is configuring the Simulation Basis Manager for the first time). After having designed the Process Plant, the mode can be changed to dynamic.

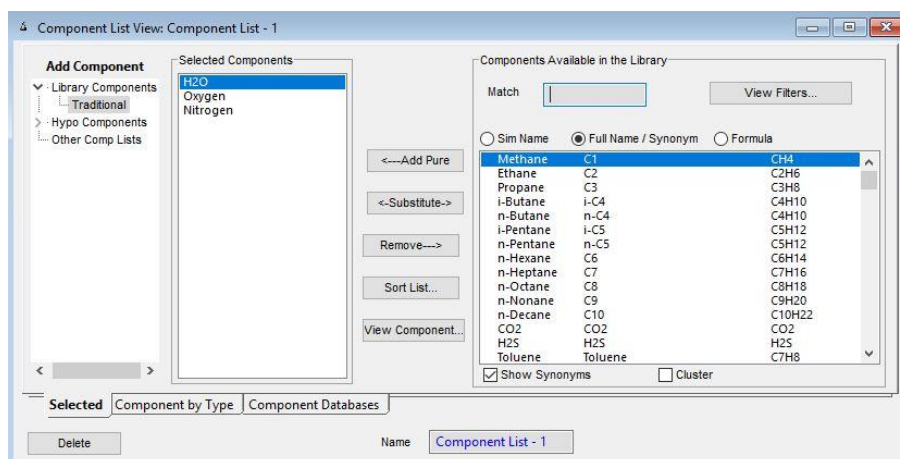


Figure 3. Component List View for the simulated case of the project

As it is seen in Figure 3, the components or chemical substances that are to be present in the Process Plant are water and air, the latter being simplified as oxygen and nitrogen.

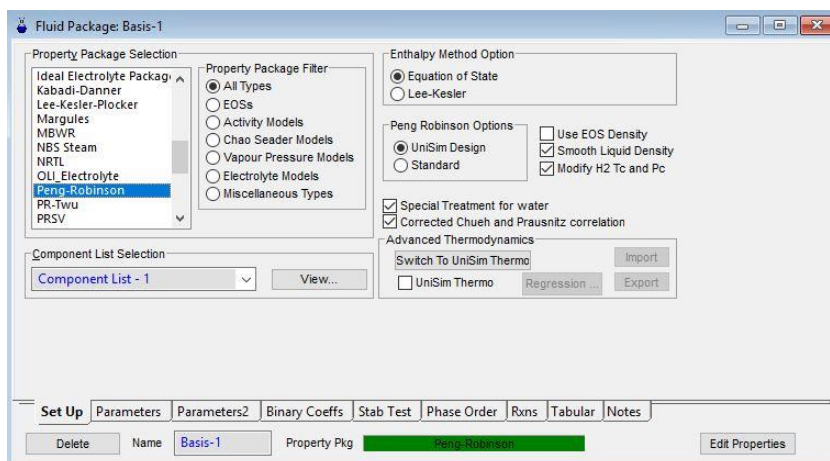


Figure 4. Fluid Package menu

As Figure 4 shows, the Fluid Package selected for the simulation corresponds to Peng-Robinson, which permits to obtain good predictions for the chemical substances previously selected. The Fluid Package can be personalized as different options are available, which can be selected by the check the box system. However, the Fluid Package has been used as per default.

1.8.2.2. Simulation Environment

After having configured the basis of the simulation, that is the chemical substances involved and the thermodynamics packages that permit it to accurately predict its behavior during the simulation, the simulation environment consists of the PFD (Process Flow Diagram) window as observed in Figure 5. A process flow diagram can be defined as a graphical representation of a chemical process, where the unit operations and the material streams are represented. Additionally, UniSim shows the control schemes involved in the process, which are usually represented together with the unit operations and material streams in other type of diagrams known as P&IDs or Piping and Instruments Diagrams.

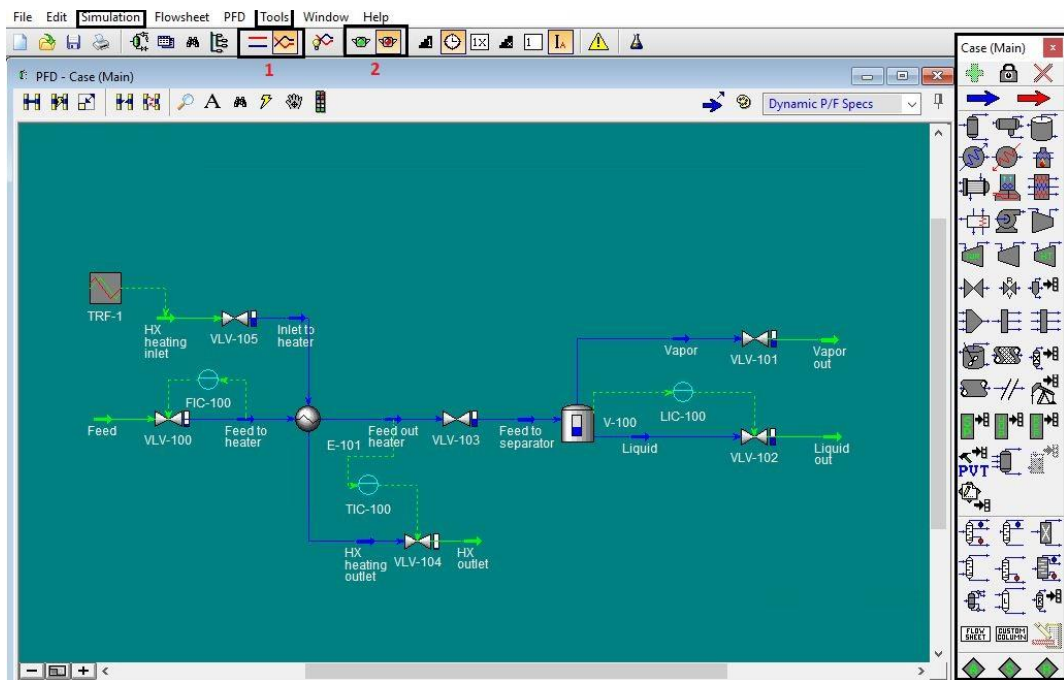


Figure 5. Simulation environment

In Figure 5, there are different elements delimited by black rectangles that must be highlighted for further explanation as follows:

- PFD window. The PFD is located inside a window at the center of the image. It permits to visualize and interact with the PFD of the Process Plant.
- Construction Palette. At the right side of the image, there is a Construction Palette, which permits to drag and drop different elements into the PFD such as unit operations, control schemes or material and energy streams for building-up the plant.
- Simulation Tab. If clicked, it is possible to select from a list the Integrator object, which is to be explained in more detail in a sub-section devoted to it.

- Tools tab. If clicked, it is possible to select from a list the DataBook object, which is to be explained in more detail in a sub-section devoted to it.
- Icons marked as (1). Interactive buttons to switch from steady-state to dynamics mode. The left one corresponds to steady-state, while one on the right corresponds to dynamics mode, which is the one selected in Figure 5.
- Green and Red Traffic Lights (2). They are interactive buttons and indicators for the present state of the simulation in a dynamic state. If the Red Traffic Light is active, as it is the case of the figure above, the simulation is stopped and it can be considered to be at steady-state. Oppositely, if the Green Traffic Light is active, the simulation is running in dynamics mode.

1.8.2.2.1 Integrator

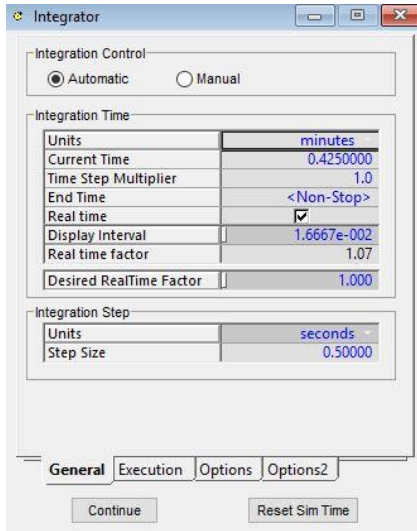


Figure 6. Integrator's General Tab

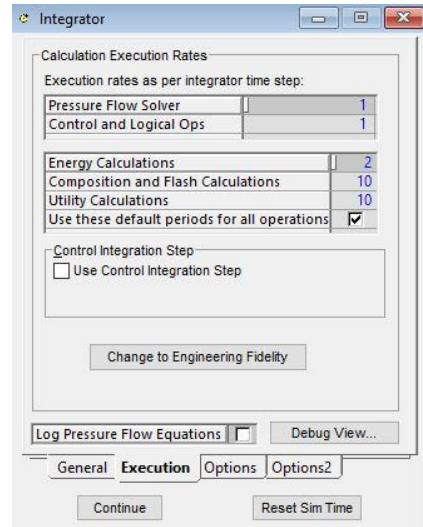


Figure 7. Integrator's Execution Tab

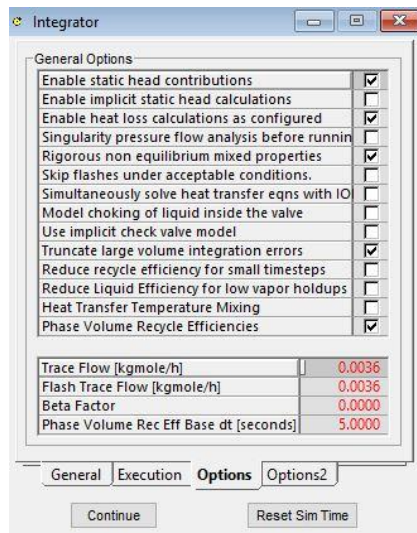


Figure 8. Integrator's Object Tab

UniSim is a simulator based on the use of Ordinary Differential Equations (ODEs) for the calculations. Thus, it uses only derivatives of one variable with respect to time. The calculations are based on integrating the derivatives over time steps defined by the user. The tinier the time steps, the more accurate the results are, but the computation demands are increased.

The Integrator object permits to configure the aforementioned integration time steps among other settings. There are three different tabs for doing so named General, Execution and Options, which are depicted in Figures 6, 7 and 8 respectively.

- General. It is possible to establish if the simulation runs in real time and if so, if an acceleration is applied to the simulation (Desired Real Time Factor). In the case of the project, the Desired Real Time Factor is set to 1, no acceleration is intended. As seen in the image, the integrator has a certain imprecision, but the Real Time Factor is maintained as close to the Desired Real Time Factor as possible.

Moreover, the step sizes of the integration time steps can be adjusted for getting the desired balance between the computational power demands and the precision on the calculations.

Finally, the Continue button allows the dynamic simulation to start.

- Execution. It is possible to establish every how many steps the calculations on Pressure-Flow equations, Control and Logical Operations, Energy, Compositions, among others are to be performed.

Variables that have major disturbances over time such as Flows require to be calculated more often than composition which does not change that tend to be constant over larger periods of time. Therefore, the Pressure-Flow Solver is recalculated every time step, while Composition Calculations are done after ten steps. No changes have been made on this tab, relying on the configuration per default of UniSim.

- Options. In this tab several options can be enabled for getting realistic results when jumping into dynamics mode. An important option that must be enabled, which is not activated per default, relates to the static head contributions. This is to account for the pressure contributed by a column of liquid below it. For example, when filling-up a water tank the increasing level of liquid exercises a pressure at any point below it. An easily understood example relates to the increase in pressure at sea when diving. It is not the same to be at the surface at atmospheric pressure than to be at ten meters depth, where there is an increase in one atmosphere and an individual is consequently submitted to two atmospheres of pressure.

1.8.2.2.2 DataBook

The DataBook serves as a data manager for interacting with all the data being generated during the simulation. It is required to use the different tabs located at the bottom of the DataBook window. Three tabs are to be commented due to its relevance in the project.

1. Variables Tab. In this tab, it is possible to open a Multi Variable Navigator that allows access to any variable from the process and select it.
2. Process Data Tables Tab. In this tab, it is possible to include in the Process Data Tables any of the variables previously selected by checking the *Include box*.

In Figure 9, all the variables seen in the image are included in a Table named *ProcData1*.

In Figure 10, the Process Data Table can be seen. Each of the variables has a *tag* associated for its identification, which in this case is based on numbers, and an *Access Mode*, which is set to

Read/Write for all the variables. This Access Mode is vital for establishing communications with third-party software as it will be commented along section 1.9.2.2. Moreover, each variable has also an *Object* associated with it, which is the element from the simulation, whose state can be described by the corresponding variable. In this project, the objects are either material streams or controllers.

As it is possible to see in Figure 10, the complete process variables set is composed of thirty-four variables in total, including pressures, mass flows, temperatures and the controllers' variables, which are SP (Set-Point Value), PV (Process Variable) and OP (Output Signal). For the main boundary material streams, which are the ones entering or exiting the process, its mass flow has been studied.

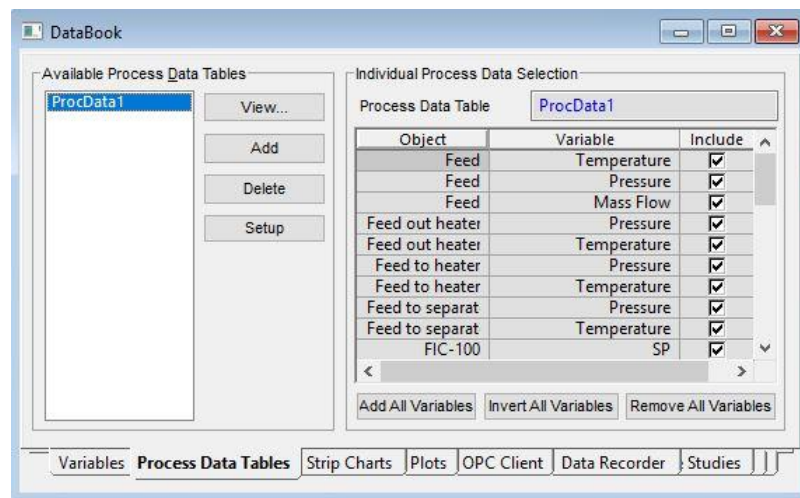


Figure 9. DataBook menu's Process Data Tables Tab

	Object	Variable	Value	Units	Tag	Access Mode
1	Feed	Temperature	24.45	C	1	Read/Write
2	Feed	Pressure	390.0	kPa	2	Read/Write
3	Feed	Mass Flow	2.849e+004	kg/h	3	Read/Write
4	Feed out heate	Pressure	130.8	kPa	4	Read/Write
5	Feed out heater	Temperature	54.43	C	5	Read/Write
6	Feed to heater	Pressure	134.5	kPa	6	Read/Write
7	Feed to heater	Temperature	24.51	C	7	Read/Write
8	Feed to separa	Pressure	111.2	kPa	8	Read/Write
9	Feed to separa	Temperature	54.43	C	9	Read/Write
10	FIC-100	SP	2.849e+004	kg/h	10	Read/Write
11	FIC-100	PV	2.849e+004	kg/h	11	Read/Write
12	FIC-100	OP	13.69	%	12	Read/Write
13	HX heating inl	Temperature	65.00	C	13	Read/Write
14	HX heating inl	Pressure	900.0	kPa	14	Read/Write
15	HX heating ou	Temperature	62.22	C	15	Read/Write
16	HX heating ou	Pressure	554.2	kPa	16	Read/Write
17	HX outlet	Temperature	62.22	C	17	Read/Write
18	HX outlet	Pressure	550.0	kPa	18	Read/Write
19	LIC-100	SP	40.00	%	19	Read/Write
20	LIC-100	PV	40.00	%	20	Read/Write
21	LIC-100	OP	44.24	%	21	Read/Write
22	Liquid	Temperature	54.43	C	22	Read/Write
23	Liquid	Pressure	109.6	kPa	23	Read/Write
24	Liquid out	Temperature	54.43	C	24	Read/Write
25	Liquid out	Pressure	101.3	kPa	25	Read/Write
26	Liquid out	Mass Flow	2.849e+004	kg/h	26	Read/Write
27	TIC-100	SP	38.00	C	27	Read/Write
28	TIC-100	PV	51.03	C	28	Read/Write
29	TIC-100	OP	100.0	%	29	Read/Write
30	Vapor	Temperature	25.00	C	30	Read/Write
31	Vapor	Pressure	101.3	kPa	31	Read/Write
32	Vapor out	Temperature	25.00	C	32	Read/Write
33	Vapor out	Pressure	101.3	kPa	33	Read/Write
34	Vapor out	Mass Flow	-0.3040	kg/h	34	Read/Write

Figure 10. Process Data Table "ProcData1"

3. Strip Charts Tab. In this tab, the user can configure the Dataloggers that permit storing data related to variables over time. The user can specify which variables to track, how many data points can be stored (Logger Size) and the frequency at which data is being collected (Sample Interval). These Dataloggers can be visualized either in tabular form (Historical) that can be saved in CSV format, or as Strip Charts.

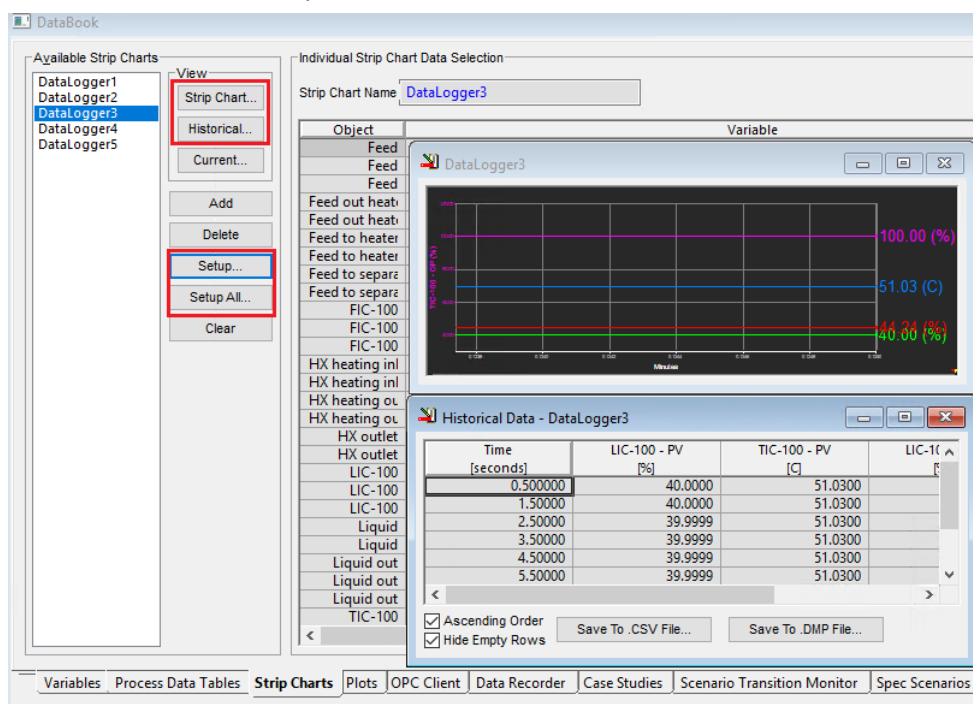
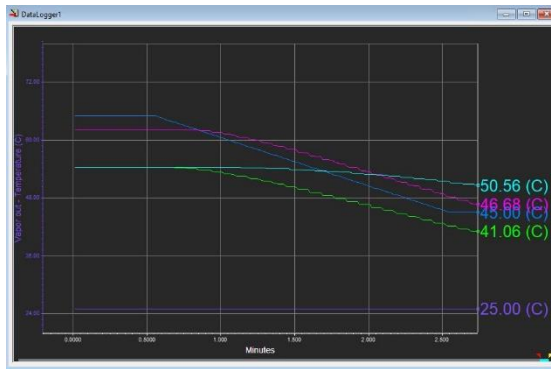


Figure 11. Databook menu for the simulated case, which includes five dataloggers.

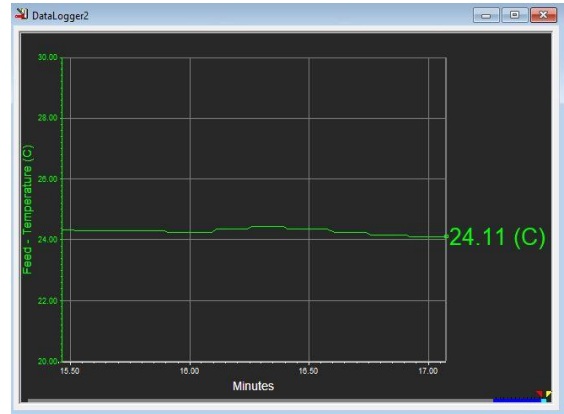
In Figure 11, it can be seen demarcated in red some of the different options available in the Databook menu. That is to say, to view a Strip Chart or the Historical, which are also depicted in the image for Datalogger3. Additionally, the options Setup and Setup All are also highlighted in red, which permit to either configure a Datalogger or all of them from the same sub-menu. All the Dataloggers configured have the same characteristics: a Logger Size of ten thousand samples and a Sample Interval of one second.

In the herein project, five dataloggers have been configured so as to study the evolution of the key variables that form part of the process variables set. In next Figures, examples for each Datalogger are depicted together with its legend for dynamic situations that can be simulated in UniSim.



Strip Chart Legend - DataLogger1

Curve Names	Colour
Feed out heater - Temperature	Green
HX heating inlet - Temperature	Blue
HX heating outlet - Temperature	Cyan
Liquid out - Temperature	Magenta
Vapor out - Temperature	Purple

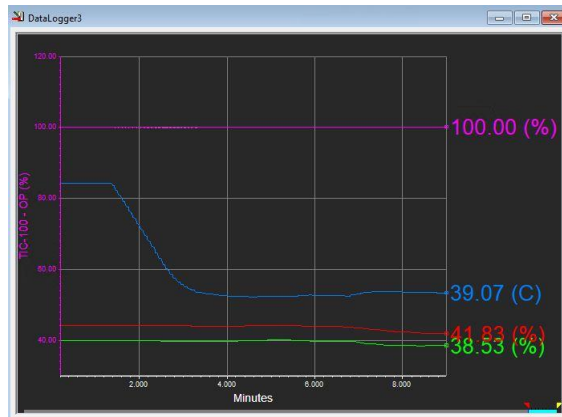


Strip Chart Legend - DataLogger2

Curve Names	Colour
Feed - Temperature	Green

Figure 12. Datalogger1 evolution after applying a negative temperature ramp on HX heating inlet.

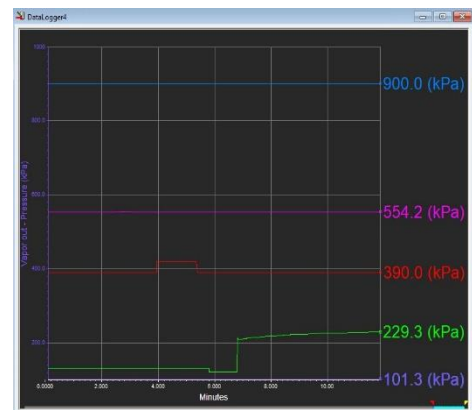
Figure 13. Datalogger2 evolution where it is observable the noise associated with the Feed Temperature introduced in the simulation as explained in Section 1.8.2.2.



Strip Chart Legend - DataLogger3

Curve Names	Colour
LIC-100 - PV	Green
TIC-100 - PV	Blue
LIC-100 - OP	Red
TIC-100 - OP	Magenta

Figure 14. Datalogger 3 evolution after the application of a negative temperature ramp on HX heating inlet and the reduction on valve opening for VLV-103 from 50 to 20%.



Strip Chart Legend - DataLogger4

Curve Names	Colour
Feed - Pressure	Red
Feed out heater - Pressure	Green
HX heating inlet - Pressure	Blue
HX heating outlet - Pressure	Cyan
Liquid out - Pressure	Magenta
Vapor out - Pressure	Purple

Figure 15. Datalogger 4 evolution after the change in pressure from 390 kPa to 420 kPa on the Feed.



Figure 16. Datalogger 5 evolution after the change in pressure from 390 kPa to 420 kPa on the Feed.

1.8.3. Description of the simulated case

The simulated case consists of a basic design, which aims to recreate a real chemical plant but simplified, that is, without considering the high level of complexity that could be found in real chemical plants. The reason for selecting a basic design is aligned with the objectives stated in the project, which only require the simulation to be rigorous, reliable and able to represent a real scenario with high fidelity. The high level of complexity found in real chemical plants could be incorporated in further designs just by adding additional details such as more streams, unit operations or feed compositions so as to accomplish new objectives imposed in future projects, without invalidating the results obtained in this project.

The actual design incorporates a feed system, with a heat exchanger intended for the regulation of the fluid temperature, that connects to a water tank open to the atmosphere, and with an output downstream.

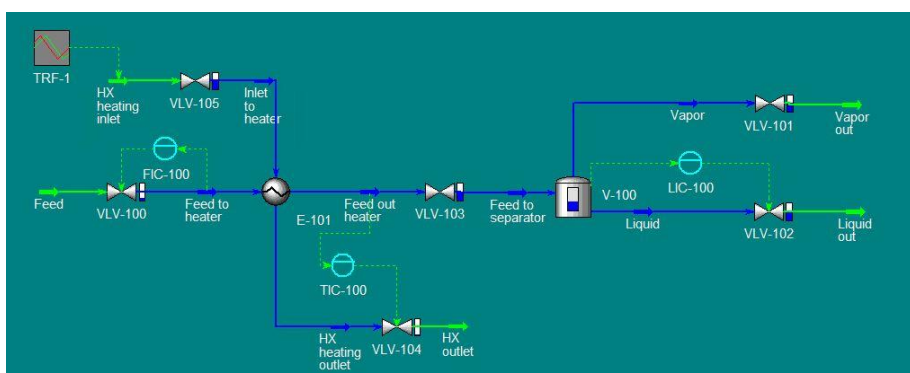


Figure 17. Process Flow Diagram (PFD) of the simulated case

The water tank has a level controller connected to the valve of the output downstream for regulating the output flow with the objective of maintaining the level at a given set point in the case of disturbances.

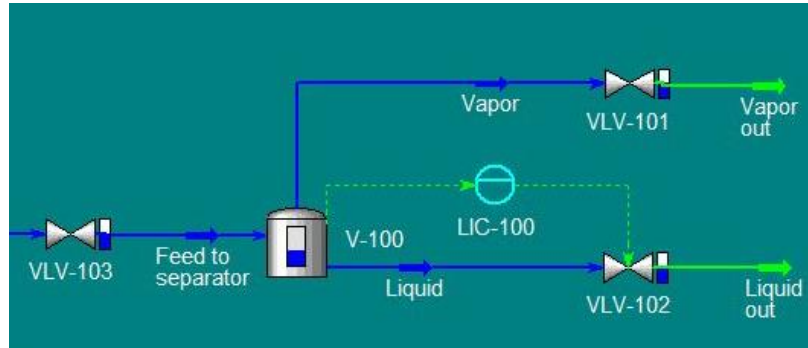


Figure 18. Water tank open to the atmosphere including a level controller in feedback configuration

The heat exchanger has a temperature controller connected to the valve in the outlet of the hot stream for regulating the amount of heating liquid required for heating up the Feed that goes to the water tank.

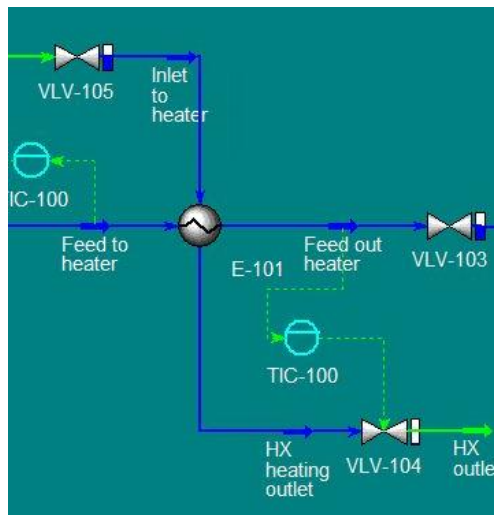


Figure 19. Heat exchanger including a temperature controller in feedback configuration

The inlet stream to the heat exchanger incorporates a Transfer Function, which permits to simulate an increase/decrease on its temperature with a ramp of $(X)^{\circ}\text{C}/\text{min}$, if activated. In other words, it is possible to specify an increase or decrease of X Celsius degrees on the temperature per each minute that passes after the activation of the aforementioned Transfer Function.

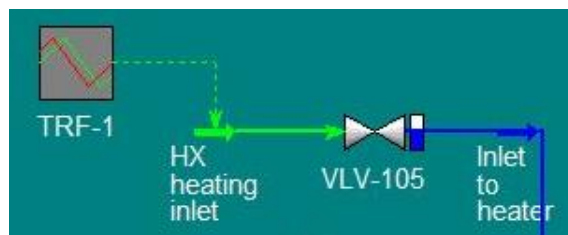


Figure 20. Transfer function connected to HX heating inlet stream

The Feed Flow incorporates a fictitious valve regulated by a Flow Controller for emulating aleatory disturbances in the Flow, which follow a normal distribution centered on the on-line value read from the simulation. The introduction of the aleatory disturbances is further explained in Section 1.9.2.2.

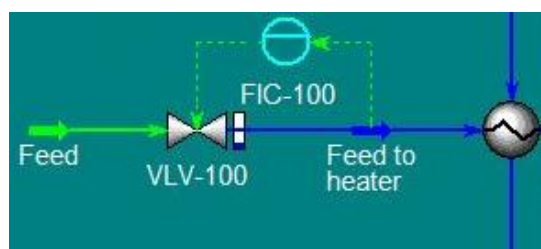


Figure 21. Fictitious valve regulated by a flow controller in feedback configuration

1.9. Communications interface programming and developing

This section covers how the communications interface for interacting with the commercial simulator UniSim has been programmed and developed.

Firstly, the programming tool employed during the project is briefly introduced and the reasons for using it are exposed. Secondly, it is delved into how the code has been developed. In the third place, the graphical user interface created for interacting with the simulation is explained. In another subsection, it is justified why part of the code written can be interpreted as a SCADA or Supervisory Control and Data Acquisition system, which stands for a software that permits the retrieval of all the information generated in a productive process.

In a final subsection, a sequence diagram in UML or Modelling Unified Language is presented for the reader to see the role the communications interface programmed has inside the big picture represented by the modular communications architecture, which is to be explained in further details in the next section 1.10.

1.9.1. Python as programming language

Python has been selected as the programming language in which to develop and program the communications interface. It could be defined as a stable programming language with a fast-growing ecosystem, being ranked among the most popular programming languages in recent years. Furthermore, its culture values open-source software, community involvement at local, national and international level, and teaching to new programmers (5) are suitable for any student interested in programming.

These values facilitate the access to the tool, contribute to the high availability of learning materials, forums (where users discuss problems in code implementation and potential solutions) and facilitate the learning curve required for acquiring the minimum skills so as to develop programming projects.

1.9.2. Python code functions for communications and monitoring

The functions that are to be commented along this section have been written in Python with the final purpose of establishing communications with UniSim, extract data from the simulated case and send it via a communications architecture for its monitoring in dynamic graphics and tables as it is to be shown in next sections: 1.10 and 1.11.

In order to understand how these functions work from a conceptual point of view without entering in the description of the code developed, two subsections are devoted to cover the main aspects behind them. In the first sub-section, it is explained how these functions have been built-up so as to interact with the simulation. In the second sub-section, it is stated the mission or objectives that each function must achieve.

1.9.2.1. Access to UniSim object's methods and properties

The Python functions coded have been developed for the direct interaction with the simulation software UniSim. This direct interaction is intended for different purposes:

1. To open or close the simulation case
2. To start or pause the dynamic simulation of the simulation case
3. To read the evolution of process variables or to impose new values to certain process variables along the time

For achieving the direct interaction with the simulation software, it has been required to access with the Python code to a type library from UniSim named *UniSimDesign.tlb* (6). This type library gathers information in binary format that can be used so as to access all the object's properties and methods exposed by UniSim at runtime (7).

On the one hand, the object's properties can be interpreted as attributes of an object or aspects of its behavior that define the object. On the other hand, the object's methods are actions that can be performed by an object (8).

The object's methods and properties exposed by UniSim have been explored with the use of Excel's Developer Environment. Concretely, the Object Browser tool has been employed to seek into the type library named *UniSimDesign*. With the aid of this tool, the user can access all the methods and properties of the different objects and see its routes. The routes refer to the group of nested instances that must be declared within the code so as to access the properties or the methods when using a programming tool such as Python.

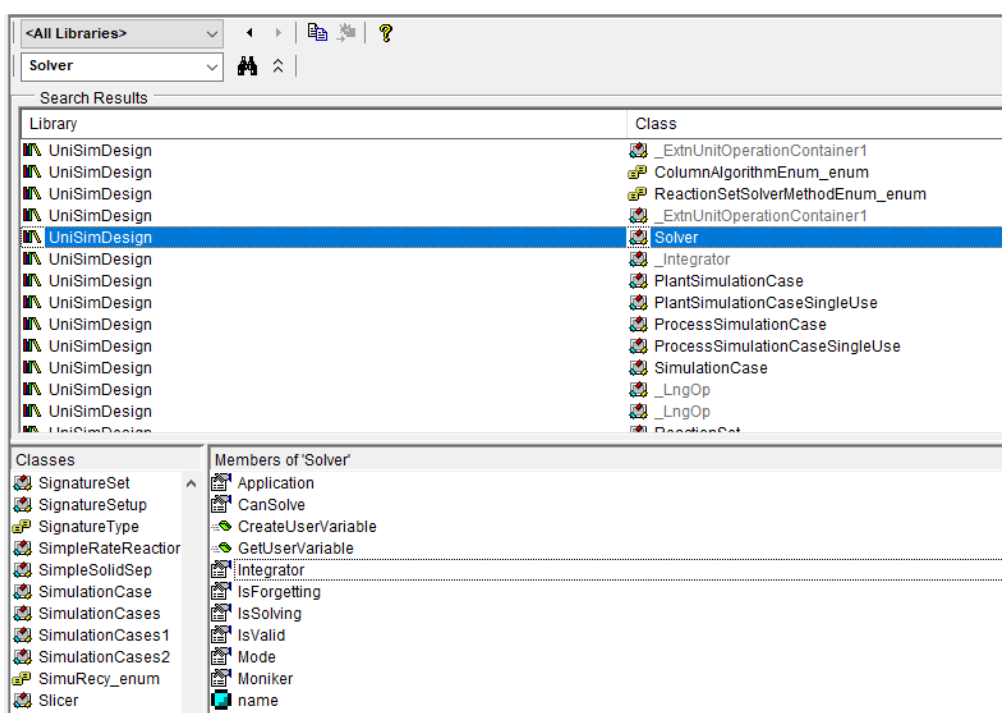


Figure 22. Search Results for the Solver object using the Object Browser tool

A search result for the Solver object from UniSim is shown in Figure 22. The Solver object includes the Integrator object, which is the tool that serves among other things for initiating or stopping the dynamic simulation of a case in UniSim. For further explanation related to the Integrator refer to Section 1.8.2.2.1.

From the image, it is concluded that the Solver object contains the Integrator object. However, it must be pointed out that all the objects are contained by the simulation case. Therefore, the route for accessing to the integrator would be:

Simulation Case -> Solver -> Integrator

If it is desired to access the properties or methods of the integrator, the route for a given method or function would be:

Simulation Case -> Solver -> Integrator -> Continue (Method to order the Integrator to initiate again the dynamics mode)

Simulation Case -> Solver -> Integrator -> GetTime (Method to get the time the simulation case has been running)

The routes that have been explained, until this point, are conceptual. When coding, these routes are constructed by the use of CLSIDs. These are unique keys that permit the Python code to identify the aforementioned objects. It could be seen as telling the code where it must search for in a language that it is capable of interpreting.

The combination of read data and gather data functions permits to have a functional Digital Twin of minor complexity when compared to the simulated case, but which permits to store the data of interest to be monitored.

1.9.2.2. Description of the functions developed

The different functions are to be described according to their mission, the objective that must be accomplished. Therefore, it is possible that for a given objective or mission, several functions are to be described as a group.

Open Case. A function that permits loading the simulated case of interest, as indicated by the user in the code, is executed.

Close case. A function that closes the simulated case is triggered.

Run Integrator. The function that calls the object's method responsible for initiating the Integrator is triggered.

Stop Integrator. It executes a function that bears a script (a sequence of instructions that indicate to stop the Integrator), which UniSim receives and the Integrator is stopped.

Read Data. A function, which accesses the information contained in the Process Data Table from UniSim indicated by the user, is initialized. In this project, the Process Data Table selected by the user is ProcData1, which is depicted in Figure 10. The information related to the numerical values, tags and units is copied in internal variables from the Python Code as long as the Access Mode is set to R (Read) or R/W (Read or Write).

Store Data. Function that performs the following actions:

1. To read data
2. Provoke noise on the data from the plant
3. Create an OPC UA Client, which can be defined as an application that can be used so as to establish communication with another software, an OPC UA Server. Once both are connected, it is possible for bidirectional communications between both so as to transfer data from the Client to the Server, or from the Server to the Client.
4. Establish communications with an OPC UA Server using this OPC UA Client and upload each of the numerical values contained in the internal variables from Python to the different tags exposed by the OPC UA Server.

1. The Store Data function calls the Read Data function. Then, the data from the Process Data Tables is copied in internal variables from the Python code.

2. The Store Data Function calls a function that introduces noise in the variables corresponding to the Temperature and the Flow of the Feed that enters in the Chemical Plant. This is done for recreating more realistic conditions, similar to the ones that can be encountered in real Chemical Plant. The noise is generated and introduced as follows:

The noise is generated as a normal distribution of 999 data points centered on 0 with a given standard deviation: for the Temperature, it has a value of 0.05°C; while for the Flow, it corresponds to 50 kg/h.

From each distribution, one value is randomly selected and summed to the value that had been read from the simulation for the temperature and the flow respectively. The resulting values are slightly different from the original values. Therefore, noise is introduced in these two variables. The slightly different values are used to overwrite the current values of Temperature or Flow in the simulation. For overwriting the values of a given variable, the variable must have the Access Mode set as W (Write) or R/W (Read or Write).

In the case of the Feed Temperature, the noise is directly introduced in the Temperature variable. Concerning the Flow, the noise is introduced into the Set Point of the controller FIC-100 connected to the valve VLV-100, which ultimately permits to simulate disturbances in the Flow.

3,4. Afterwards, the rest of code from The Store Data function permits uploading the data to the OPC UA Server. Once the data is sent to the Server, it will be sent via messages from the Server through the rest of the modular communications architecture, whose modules are to be described in detail in the next Section 1.10.

The execution of the Store Data function occurs in a loop that is repeated automatically. The function stops when the User pushes the “Stop Integrator” button that will be introduced and briefly described in Section 1.9.3. When the button is pressed, the Integrator stops and the uploading of data to the OPC UA Server is ended.

1.9.3. Built-in interface (GUI library Tk)

A graphical user interface can be defined as a form of user interface that allows users to interact with electronic devices through graphical icons, instead of text-based user interfaces, typed command labels or text navigation (9).

The nature of the code developed, which is intended for the continuous interaction with the simulation, demands the execution of individual functions at specific timing. For this reason, it has been decided to develop a GUI in Python for the user to be able to execute the functions developed in the order chosen by them at will and at specific timing.

A graphical user interface (GUI) has been developed with the standard GUI library for Python named Tkinter or commonly abbreviated as Tk. The library permits you to create a GUI application fast and easily. A GUI application consists of a main window, which can incorporate widgets such as buttons, labels and text boxes among others. It is refreshed constantly according to a main event loop (10).

The buttons are interactable objects that when pushed, they trigger the execution of the function/s they are associated with. For the GUI developed, each of the buttons is labelled according to the global mission achieved by the functions that are triggered when the button is executed.

In figure 23, it is possible to see the buttons corresponding to functions developed for communications and monitoring, which are **Open Case**, **Close case**, **Run Integrator**, **Stop Integrator** and **Store Data**.

INTERFACE TO UNISIM

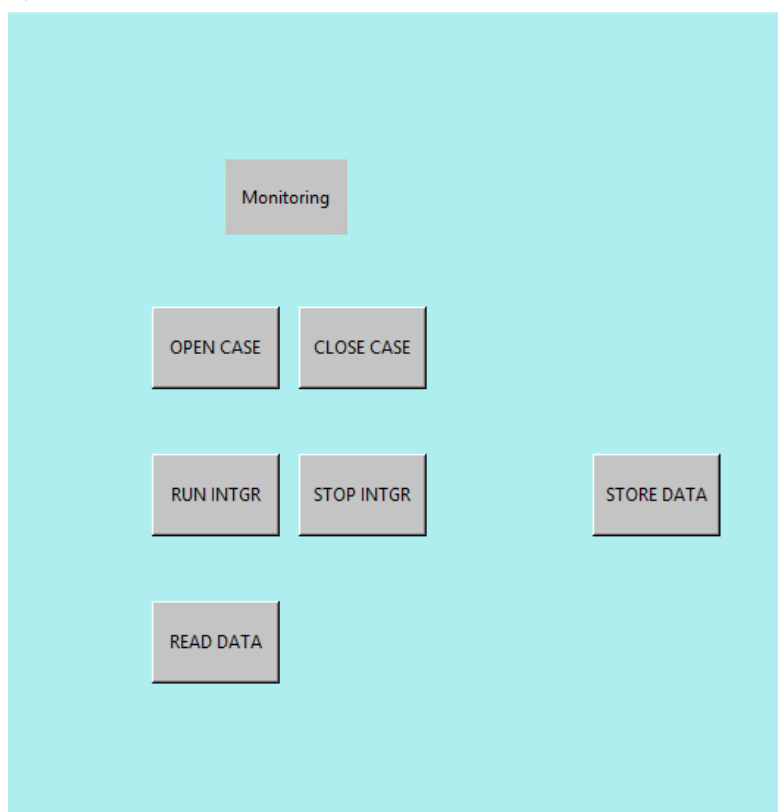


Figure 23. Graphical User Interface for Monitoring and Communications

1.9.4. Role of the corresponding code as SCADA

The button named **Store Data** from the GUI previously described, triggers several functions of the code, which read data from the simulation and gather it in internal variables. The data read and stored could be seen as the data recorded by the different sensors installed in the Chemical Plant such as temperature and pressure probes, flowmeters, etc. Afterwards, the values stored are sent via a standard communications protocol for feeding a modular communications architecture.

As commented at the beginning of Section 1.9, the actual definition for SCADA stands for a software that permits the retrieval of all the information generated in a productive process. When pushing the button known as Store Data, a piece of code, which can be seen as a software, gathers the information related to the different plant sensors and shows them for its subsequent routing via standard communications protocol. As a result, it recreates the behavior of any industrial SCADA as per the previous definition.

1.9.5. Sequence Diagram for communications and monitoring in UML standard

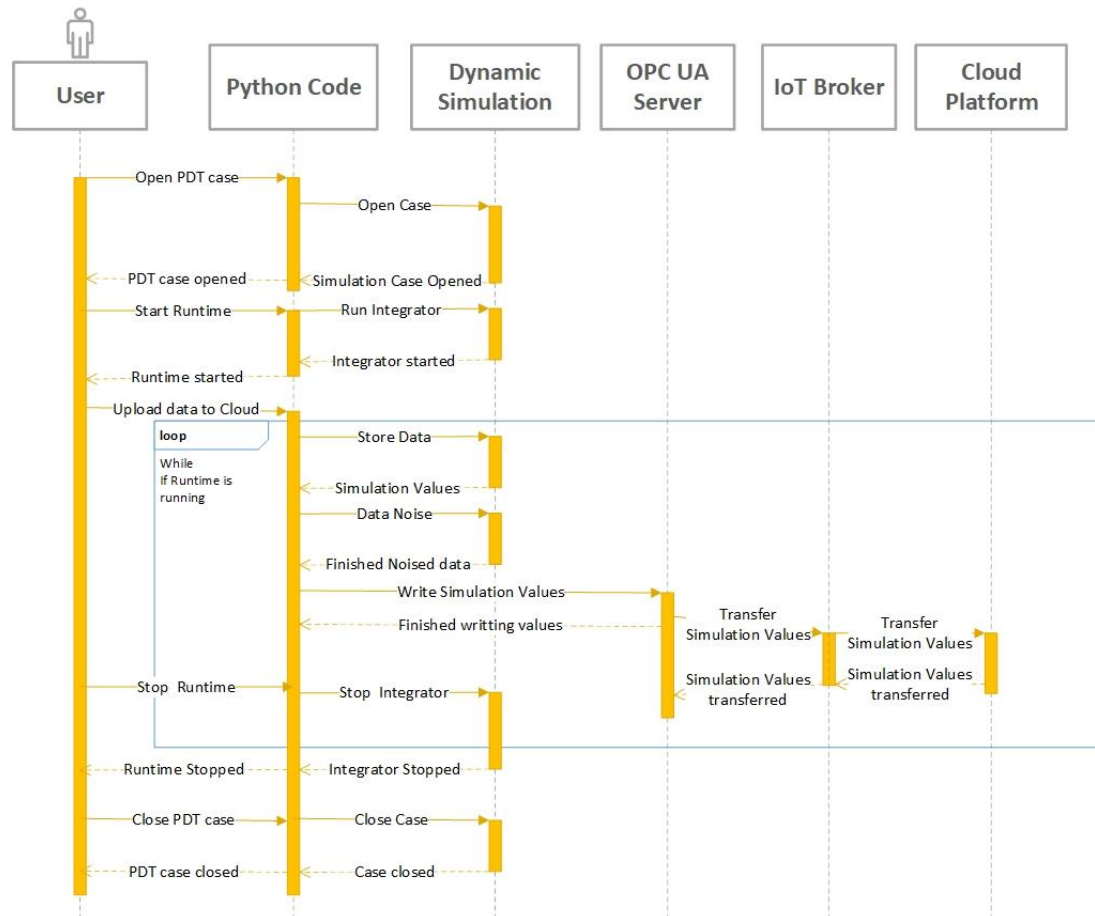


Figure 24. Sequence Diagram for communications and monitoring in UML

1.10. Modular communications architecture

1.10.1. Modular architecture

The communications systems that have designed consist of a modular architecture in the stage of prototype as it is to be justified afterwards. The modules employed are:

1. Process Plant in the form of a simulation case that represents a chemical plant, that is a Digital Twin.
2. A SCADA that uses OPC UA Communications consisting of an OPC UA Client, an OPC UA Server, which have already been introduced, and an IoT Broker. The Broker is a software that permits the exchange of data from a data source such as an OPC UA Server to a destination source such as a Cloud Platform.
3. Cloud Platform that includes many services for data monitoring, analysis and treatment, among others.

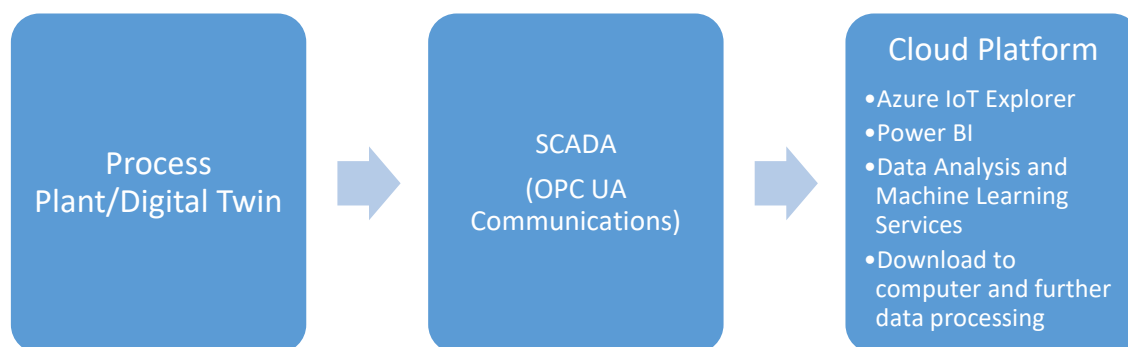


Figure 25. Block Diagram depicting the Modular communications architecture, where each module contains a list of its constituents

On the one hand, the architecture is modular since each of the modules that conform it can be substituted for an equivalent module depending on the user's situation, interests or possibilities. For instance, if the user wishes to make use of another communications protocol for the Server or the same protocol but with a Server from another company, the modular architecture proposed would still be functional. As long as the logics and algorithms that constitute the architecture are maintained, other equivalent modules could be:

- The substitution of the process simulation tool by a real process plant
- The use of another Cloud Platform

On the other hand, the architecture is in prototype stage since it consists of the minimum modules with the minimum capabilities required to accomplish the objectives of the project. It is a DEMO for showing the viability associated with the ideas that have been explored and the potential associated with future ideas linked to this research field.

1.10.1.1. Process Plant

The first module corresponds to the Process Plant, which has been built-up in the form of a simulation case developed with UNISIM® Design Suite R480, conforming what is known as a Digital Twin, which has been previously described in section 1.8.3.

1.10.1.2. SCADA (OPC UA Communications)

A SCADA has been implemented in Python as it is described in sections 1.9.2 and 1.9.4. The SCADA transmits information to the modular communications architecture with the use of OPC UA Communications.

OPC UA stands for OPC Unified Architecture. It is an extensible, platform-independent standard that enables the secure exchange of information in industrial systems. It was released by the Open Platform Communications (OPC) Foundation in 2008.

OPC UA is compatible with Windows, macOS, Android, and Linux. It can also be used in embedded systems and bare-metal systems, which do not use an operating system. OPC UA works on PCs, Cloud-based infrastructures, PLCs, micro-controllers, and cyber physical systems (CPS).

The goal of OPC UA is to enhance interoperability between hardware devices, and enterprise planning and automation software by providing a framework for industrial companies to converge disparate technologies.

Related to the security of the transmissions, OPC UA enables data encryption at the data source, ensuring secure transmission without relying on network firewalls at the system's core. This means security is ensured from the start of the data's transmission (11).

This communications protocol has been selected due to its widespread use in industry and its high compatibility with all the existing platforms.

The first aspect permits to perform a project in the same conditions as those encountered in the industry, which would facilitate the adoption, at industrial scale, of the modular communications architecture developed.

The second aspect permits to increase the quantity of modules that can be assembled in the communications structure, which permits to exploit the property of modularity the architecture possesses. For instance, if company A desires to work on Windows, while company B wants to work on macOS, the same architecture works in both scenarios. Another example is related to the final destination of the data. The architecture proposed is intended for sending data to the Cloud. However, if a company prefers to send the data to other computers, maintaining the data at local level, the architecture also works.

The OPC UA Communications developed in this project are used by three different sub-units: an OPC UA Server (an independent software), an OPC UA Client (an application programmed in the Python code) and an IoT Broker (an independent software).

1.10.1.2.1 OPC UA Server

All the information related to the installation of the software, the configuration of the server and the CSV simulation files is based on the information provided in the Quick User's Guide for the configuration of the OPC UA Server from the company Integration Objects, which is available at (12), requiring a prior free of charge registration.

Installation of the software

The software installed is a DEMO that can run during 48 hours without interruption, after which it must be initiated again for additional periods of 48 hours. Therefore, the DEMO serves for testing the communications architecture proposed in this project, which does not need to be functional for 48 hours in a row. Notwithstanding, if future projects, requiring more time of execution for the communications architecture, are to be developed, other alternatives should be assessed such as searching for other kinds of DEMOs or paying a license for a software that could run without restrictions.

Configuration of the server

The server can be configured from the settings tab which include menus for *Security Policies* and *Configuration* respectively.

In the menu *Security Policies*, it is possible to establish the kind of security mode to be associated with the OPC UA Server Simulator. Once an OPC UA client tries to access the Server, it should use one of the secure channels available. It is possible to impose the obligation to Sign with a Username and a Password or to access with no credentials. Moreover, data to be transferred from the Client to the Server or from the Server to other reception points can be encrypted and also protected by the TLS protocol. Finally, it is possible to specify the endpoint to be TCP, HTTPS or both.

In the menu *Configuration*, it is possible to configure the port numbers and the server names for TCP and HTTPS connections, which compose the *endpoint* that allows to identify the Server. Additionally, it is possible to set the update rate in milliseconds, which is the frequency at which the data that is simulated by the server gets refreshed.

CSV Simulation Files

The OPC UA Server Simulator uses 2 CSV simulation files that can be found inside a folder named DATA located at the installation folder. Both of them can be modified so as to personalize the Server according to the needs of the user:

- “AddressSpace.csv” used to build the address space of the OPC UA Server. It is composed of columns named *Tag Name*, *Data Type*, *AccessRights*, *Simulated*. Additional tags can be added just by filling a new row in CSV format and specifying the aforementioned columns:

Tag Name: The name of the new tag. A tag can be defined as a container in which it is possible to store a piece of information temporally.

Data Type: The type of data supported (Int16, Int32, Double, String, Boolean among others) by the tag

AccessRights: The Access Rights the OPC UA Client has, which can be RW: Read and Write, R: Read, W: Write

Simulated: True if the data is being simulated by the software or False if data is being introduced by an OPC UA Client.

Tag Name	Data Type	AccessRights	Simulated
Tag1	IO_Int16	RW	FALSE
Tag2	IO_Int32	RW	FALSE
Tag3	IO_Int64	RW	FALSE
Tag4	IO_UInt16	RW	FALSE
Tag5	IO_UInt32	RW	FALSE
Tag6	IO_UInt64	RW	FALSE
Tag7	IO_Double	RW	FALSE
Tag8	IO_String	RW	FALSE
Flow	IO_Byte	RW	FALSE
Tag10	IO_Boolean	RW	FALSE
Tag11	IO_Int16	R	TRUE
Tag12	IO_Int32	R	TRUE
Tag13	IO_Int64	R	TRUE
Tag14	IO_UInt16	R	TRUE
Tag15	IO_UInt32	R	TRUE
Tag16	IO_UInt64	R	TRUE
Tag17	IO_Double	R	TRUE
Tag18	IO_String	R	TRUE
Tag19	IO_Byte	R	TRUE
Tag20	IO_Boolean	R	TRUE

Figure 26. Example of AdressSpace.csv

- “ValueSpace.csv” used to simulate the data values of the OPC UA items. The data values are simulated in cycles. For instance, if it is specified the simulation of 10 values for a given tag, once the 10 values have been simulated the cycle ends. Then, a new cycle begins.

The csv file contains two different kinds of columns for each tag. The first kind contains the values that the tag being simulated will have and its column heading is the *Tag Name*; the second kind has no column heading but serves to indicate the connection status.

1.10.1.2.2 OPC UA Client

The values associated with the different variables are extracted from UniSim and stored in internal variables of the Python code. Then, the SCADA sends the variables' values via OPC UA communications with an OPC UA Client programmed in Python. The Client sends all the values in the form of a telemetry message to an OPC UA Server. The Server has been personalized so it contains a tag for each internal variable from the Python code. Each tag has been configured with *AccessRights* of Read/Write, the *Data Type* of Integer and the *Simulated* option as False. This process is repeated in a loop until the user clicks on the "Stop Integrator" button that stops the simulation and the transference of data.

In the Python code, it is programmed for each internal variable to point out towards a unique tag so that each tag is associated with a unique variable in the telemetry messages. As well, the OPA UA Client is also programmed with the characteristic endpoint that identifies the OPC UA Server at which it sends the telemetry messages. Since the Server configured permits access with no credentials, the OPC UA Client programmed in Python access to it anonymously.

It is important to differentiate two types of variables when sending them through OPC UA communications. There are variables that are constant, while others vary over time. OPC UA communications refresh tags when the information received in the telemetry messages vary over time. Therefore, the tags with variables that are constant only receive the first telemetry message and are no longer refreshed. If not corrected, the given tag's information is not refreshed and is not sent to the Cloud.

For a tag to be refreshed when associated with a constant variable, it has been decided to introduce in the Python code some lines of code that do the following when reading constant variables:

- For odd readings, constant value+0.000000001
- For even readings, constant value-0.000000001

Therefore, the value received is constantly changing and the value sent via OPC UA Communications does not suffer significant modifications with respect to the constant value, since the summations and subtractions are negligible.

1.10.1.2.3 IoT Broker

All the information related to the installation of the software and the configuration of the program DataFEED OPC Suite is based on the information provided in the DataFEED Quick User's Guide from the company Softing, which is available at (13), requiring a prior free of charge registration.

Installation of the software

The software DataFEED OPC Suite installed is a DEMO that runs for 72 hours in a limited demonstration mode. During this period all the features are completely enabled, but after it the program stops all functionalities. Restarting the software will start a new 72-hours demonstration period.

Configuration of the IoT Broker

The software DataFEED OPC Suite includes many services, among which are included the options to connect to OPC UA Servers, to configure a MQTT Broker and to exchange information between the OPC UA Server and the MQTT Broker. The services are grouped according to its functionality with regards to data, hence services can be for *Data Source*, *Data Processing* or *Data Destination*.

OPC UA Server

The Service OPC UA Server can be found inside the group *Data Source*. Once in the OPC UA Server menu, the option *Add a new data source* is clicked. A menu for configuring an OPC UA Server pops up. The important information to provide in the menu can be summarized as follows:

- Connection Settings: Connection name. The name by which Softing DataFEED OPC Suite will recognize the Server once the connection to it is created.
- Endpoint Settings: OPC UA Server Endpoint. The endpoint URL is provided. In the case of this project, the endpoint corresponding to the OPC UA Server from Integrations Objects is provided.
- Security Settings: The way of requesting connection to the Server is indicated. That is to indicate if the request should be made with or without credentials, to trust Servers Certificates. In the case of this project, the connection is performed without credentials and the option *Accept all Servers Certificates automatically* is enabled.

MQTT Broker

A MQTT Broker transforms all the tags from the OPC UA Server, which represent the Digital Twin status, into an IoT Device. Subsequently, the information contained in this IoT Device is sent in the form of telemetry messages towards Azure IoT Hub.

Azure IoT Hub is a service from the Cloud Platform of Microsoft Azure that serves as a message center for the bidirectional communications between an IoT Device and a device hosted in the Hub (14). Therefore, it is possible to send the data of the Digital Twin to a device hosted in the hub so as to have the data introduced in the Cloud. In section 1.11, it is explained what to do with the data that is introduced in the Cloud.

The Service MQTT Broker can be found inside the group *Data Destination*. Once in the MQTT Broker menu, the option *Add a new data source* is clicked. A menu for configuring a MQTT Broker pops up. The important information to provide in the menu can be summarized as follows:

- Connection Settings: Connection Name. The name by which Softing DataFEED OPC Suite will recognize the MQTT Broker once it is created.
- Connection Settings: Client ID. The name of the device hosted at the Azure IoT Hub to which Softing DataFEED OPC Suite will connect.
- Communication Settings: MQTT Broker URI. The URI has the following structure:
 - `ssl:// Name of the Azure IoT Hub.azure-devices.net: Port number`
- Communication Settings: Authentication Settings. It must be provided the following:
 - User Identity. Username and password option is selected.
 - User Name. It has the structure:
 - **Name of the Azure IoT Hub**.azure-devices.net/Client ID
 - Password. It corresponds to the *SAS token connection string* for the given Client ID to which it is desired to establish connection. This password is obtained with an auxiliary program, which is explained in detail at Section 1.10.1.3.1.

Once the Communication Settings have been filled, a *Connection Test* should be performed so as to assess if the connection is successful or not.

Communication Settings

On this wizard page the communication settings of the data destination connection to an external MQTT Broker for publishing data are configured.

Connection: MQTT_Azure_Local

MQTT Broker Settings

MQTT Broker URI

< IP address or hostname with domain of the broker > [: <port number >]

ssl:// IoTHubTFMDIVP.azure-devices.net:8883

MQTT Broker Certificate

Select the MQTT Broker certification handling mode.

Accept all Servers Certificates automatically

Accept Trusted Certificates, only

Authentication Settings

User Identity: User name and password Use Client Certificate

User Name: IoTHubTFMDIVP.azure-devices.net/Local_Agent

Password:

Connection Test

Use the button on the right side in order to test the connectivity to the corresponding MQTT broker.

Figure 27. Example of a configured Communication Settings

- MQTT Topic definition. On this wizard page the address space for the current connection can be defined. It consists of a nested structure that contains the device from the Azure IoT Hub for which the connection has been configured. The device receives messages in the form of events. It is possible to add new items in the form of events that the device will receive. It will be added as many items as tags from the OPC UA Server are intended to be sent to the Cloud.

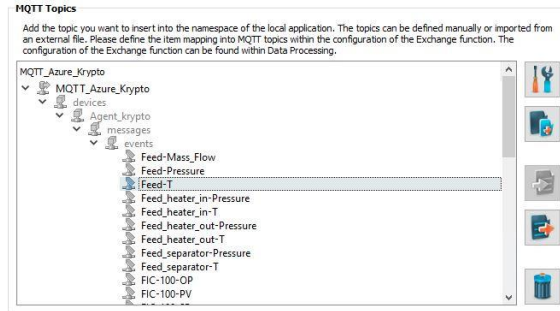


Figure 28. Example of nested structure for a connection to an Azure IoT Hub

Once an item is added, it must be configured as follows:

- Topic Class. Tag is selected in the herein project
- Name. The name to identify the given Tag for when configuring the exchange of information between the OPC UA Server and MQTT Broker.
- Publish Format. Several formats such as JSON among others are available. However, the option *User defined format* is selected so as to personalize the tags sent to Microsoft Azure for its future processing in the Cloud Platform. Specifically, the defined format consists of a JSON format incorporating the timestamp, the value, the quality and the name for each tag.

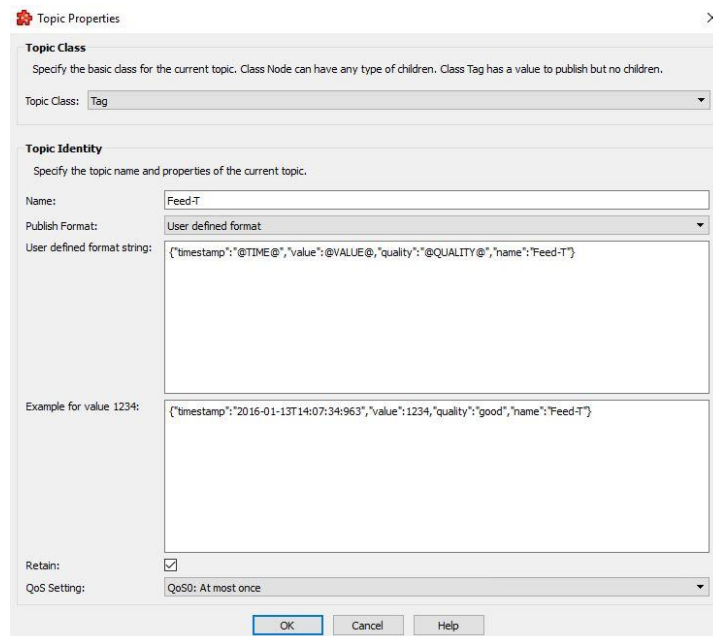


Figure 29. Configuration of an item according to the User defined format

Exchange of information between OPC UA Server and MQTT Broker

The exchange of information service can be found inside the group *Data Processing* labelled as *Exchange*. It permits to select a resource acting as data source (OPC UA Server) and a resource acting as data destination (MQTT Broker). In order to exchange information between the resources, it is necessary to:

1. From each of the resources, unfold its nested data structure. On the one hand, the Server contains several tags. On the other hand, the MQTT Broker was configured to send data towards a device, which received that data in *events*. The aforementioned *events* is nested and composed of different tags.
2. Having the nested structures unfolded, it is possible to select the desired tag of the Server to be sent (source) and the tag of the MQTT Broker, which acts as the receiving point (destination).
3. After both source and destination are selected, the option *Use the selected items as a new Exchange action* is clicked.
4. At the Exchange - Source Items Data List, the source and destination items selected are displayed. It is important to remark that additional information is displayed such as the Update Rate. The latter must be configured according to the speed at which the OPC UA Server gets refreshed with new data.

Source Item	Destination Item	Update Rate	Condition	Ex. Value(s)
OPC-UA_Server.1.Real Time Data.Tag2	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Feed-T	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag22	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Feed_separator-Pressure	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag23	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Feed_separator-T	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag3	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Feed-Pressure	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag38	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Liquid_out-T	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag39	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Liquid_out-Pressure	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag40	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Liquid_out-Mass_Flow	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag41	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.TIC-100-SP	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag42	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.TIC-100-PV	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag43	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.TIC-100-OP	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag46	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Vapor_out-T	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag47	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Vapor_out-Pressure	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag48	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Vapor_out-Mass_Flow	12000 ms	Alias (Always)	Alias (Data Value)
OPC-UA_Server.1.Real Time Data.Tag49	MQTT_Azure_Krypto.devices.Agent_krypto.messages.events.Mahalanobis_distance_mean	12000 ms	Alias (Always)	Alias (Data Value)

Figure 30. Configuration for the Exchange Information between the OPC UA Server and the MQTT Broker. The Update Rate has been imposed as of 12 seconds in this project for the synchronization with the OPC UA Server.

Why is it important to configure the Update Rate?

The Update Rate is the time in milliseconds, after which the IoT Broker connects to the OPC UA Server and requests data. This request of information must be synchronized with the Data Writing performed by the OPC UA Client. That is to say, it is necessary to establish an Update Rate high enough for the OPC UA Client to have refreshed the data on the OPC UA Server.

For instance, if the OPC UA Client requires 10 seconds to refresh data on the OPC UA Server, an Update Rate of 12000 milliseconds can be established.

Having configured the Exchange of information between the Server and the IoT Broker, it is possible to initiate the transfer of information from the OPC UA Server to the Cloud by Starting the software DataFEED OPC Suite from the tab *Local Application* -> *Start*.

1.10.1.3. Cloud Platform

1.10.1.3.1 Azure IoT Explorer

Azure IoT Explorer is a graphical tool for interacting with devices connected to a given Azure IoT Hub: to monitor and manage them. However, it is used as an auxiliary program in this project as it has been stated in Section 1.10.1.2.3. It is employed for granting access to the Azure IoT Hub of interest for a given amount of time. The software belongs to Microsoft and it is installed as a free of charge software (15).

Configuration of the software

In the menu Home, it must be entered inside the sub-menu *IoT hubs*. In the sub-menu, a new connection corresponding to a given Azure IoT Hub is added. It is necessary to introduce the Connection String of the Hub. After that, the connection is created and it is possible to access the Hub from Azure IoT Explorer.

Then it is possible to interact with the devices from the Hub clicking on *View devices in this hub*. Once inside the device menu, there are different options that can be explored related to monitoring and management of the device. Focus will be given to the *Device identity* menu. There is a drop-down option named *Connection string with SAS token*. If unfolded, it is possible to generate a SAS token connection string based on the Primary key and valid for the quantity of time (minutes) specified by the user.

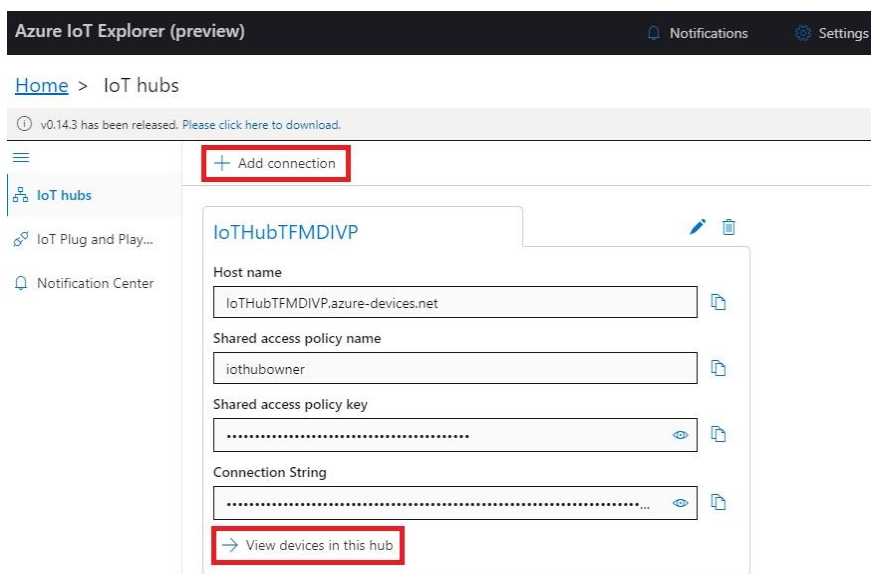


Figure 31. Sub-menu IoT hubs from which it is possible to add new connections and to access devices of already configured hubs.

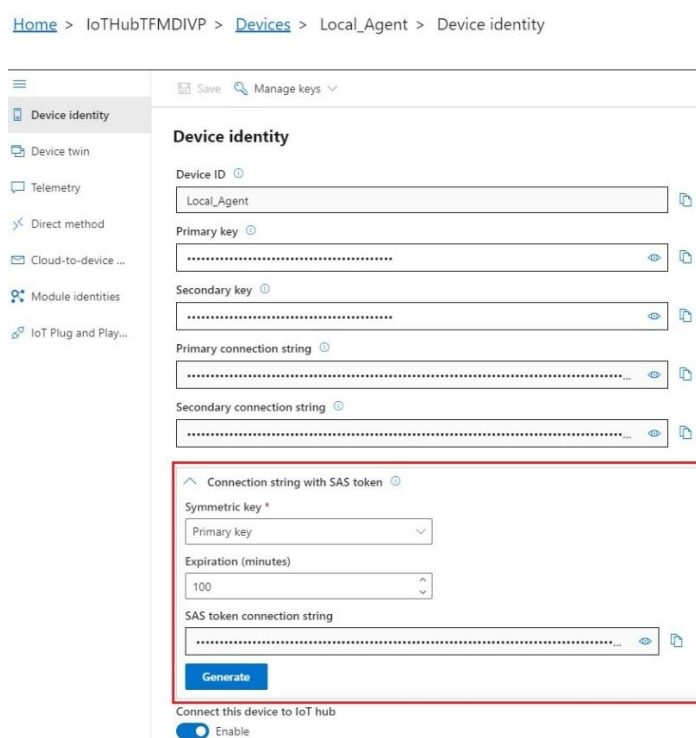


Figure 32. Device identity menu for a given device. The drop-down option named Connection string with SAS token delimited by a red square appears unfolded and already configured.

1.11. Publication of on-line data through the internet

1.11.1. Modular architecture

The communications architecture proposed to be used in the Cloud consists of services that can be connected for enhancing their individual capabilities. Therefore, each of these services can be considered as a module, the sum of which comprises a modular communications architecture Cloud-hosted. The architecture assessed in the project can be divided in three modules or services: ingest, analyze and deliver data respectively as it is depicted in Figure 33.

Firstly, the IoT Device created by the IoT Broker sends the Data to a device hosted in the service Azure IoT Hub. The data is received by the device in the form of telemetry messages. This stage could be seen as Ingesting the Data.

In second place, the telemetry messages received by the device are directed to the service Stream Analytics, which sends them to Power BI. While being handled by Stream Analytics, the telemetry messages are pre-processed with a piece of code written in SQL language. This pre-processing consists of gathering in a table the information contained in the different telemetry messages as a function of the variable they carry information about, where cells correspond to all the variables received for a given reception time and columns correspond to the respective variables and the reception time. After the pre-processing, the data arranged in tabular form is sent to Power BI. This stage could be seen as Analyzing the Data.

Finally, the data received in Power BI can be displayed in dynamic tables and graphics. It is also possible to establish an alarm for a variable of interest when it surpasses or drops down a threshold. On the other hand, data from Power BI can be downloaded to a personal computer such as in the form of csv or excel files. This stage could be seen as Delivering the Data.

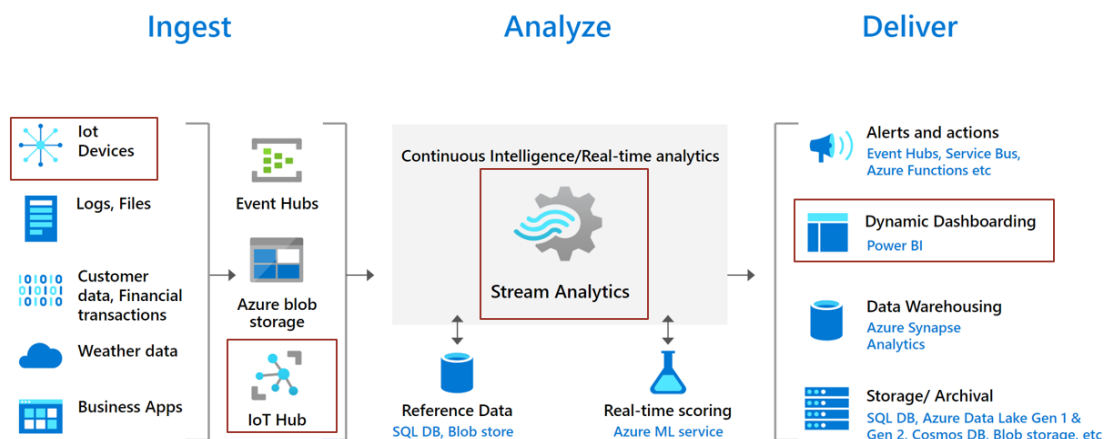


Figure 33. Different services available at Microsoft Azure grouped by Ingest, Analyze or Deliver data extracted from (16).

In figure 33, it is possible to see the different stages that have been described previously: ingest, analyze and deliver the Data. The services offered by Microsoft Azure Cloud are quite extensive as it is appreciable in the image.

In this project, the services employed are those, which allow to constitute a modular architecture in the stage of prototype for the same reasons as those exposed when describing the communications structure used for sending information to the Cloud. Specifically, the services used are IoT Hub that reads from an IoT Device in the stage of Ingest, Stream Analytics in the stage of Analyzed and Power BI in the stage of Deliver as it is indicated with red boxes in Figure 33.

In the next subsections, it is intended to briefly describe each of the services used at the Cloud Platform. That is to give a brief grasp on the theory behind each of the services, how they have been configured and the results that are obtained after their use. Besides, a sub-section is devoted to how to connect them. In other words, how it is possible to establish a message route for sending messages from an IoT Device to Power BI, using as intermediates IoT Hub and Stream Analytics. Finally, there is a sub-section where the advantages and disadvantages related to the use of the Cloud are summarized.

1.11.2. Azure IoT Hub

Azure IoT Hub is a Cloud-hosted service that permits the bidirectional communication between IoT devices and Azure. It is based on the use of device-to-Cloud telemetry data.

On the one hand, it is possible to understand the state of the devices connected, reliably send commands and notifications to the connected devices, and track message delivery with acknowledgement receipts. On the other hand, the messages received by the devices can be routed to other Azure services by defining message routes (17).

In this project, the service has been deployed in its free tier version, which is intended for testing and evaluation. It allows 500 devices to be connected to the hub and up to 8,000 messages per day. The service has been used for routing the messages, received by a device hosted in Azure IoT Hub, to Stream Analytics which is an additional Azure service.

The service has been deployed by following the instructions available at (18), which also explain, among other topics, how to create a device. Regarding how to send messages, the message route has been configured according to the guidelines provided at (19).

1.11.3. Stream Analytics

Stream Analytics is a Cloud-hosted service with which it is possible to build streaming data pipelines using SQL. It is easily extensible with custom code and built-in machine learning capabilities for more advanced scenarios (20).

A streaming data pipeline enables the smooth, automated flow of information from one point to another. It prevents many of the common problems that an enterprise can experience such as information corruption, bottlenecks, conflict between data sources, and the generation of duplicate entries.

Streaming data pipelines, by extension, are data pipeline architectures that handle millions of events at scale, in real time. As a result, it is possible to collect, analyze, and store large amounts of information. That capability allows for applications, analytics, and reporting in real time (21).

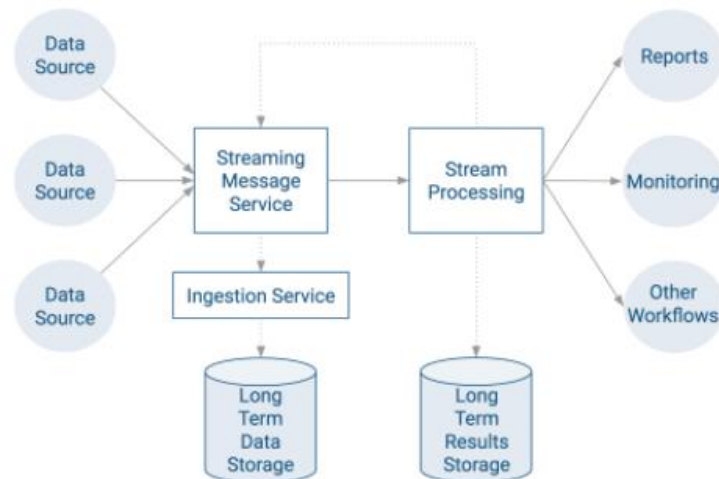


Figure 34. Data Streaming Pipeline Architecture extracted from (22)

In the project, the focus on the use of this service has been made on the reporting capabilities offered. Therefore, the service has been deployed as it is instructed at (19), and a query (a set of orders) has been coded in SQL for routing the messages from Azure IoT Hub to Power BI. Prior to its reporting, the data has been pre-processed with SQL for its display in tabular form.

The telemetry data received at Stream Analytics consists of numerical values identified by tags among other properties such as the timestamp or the status of the message. As it is possible to see in Figure 35, if this data was to be sent to a Power BI table directly with no pre-processing applied, it would be sent in tabular form to Power BI having as columns: timestamp, value, name, etc. Therefore, numerical values would be all grouped together with no distinction depending on the tag, which does not allow to display a given tag as a function of time in a graphic.

Input preview [Test results](#)

Showing events from 'DataIoTHub'. This list of events might not be complete. Select a specific time range to show all events during that period.

View in JSON Table Raw Refresh Select time range Upload sample input Download sample data

timestamp	value	quality	name	EventProcessedUtc...	PartitionId
"2021-06-15T01:36:55....	101.3000000000001	"good"	"Liquid_out-Pressure"	"2021-06-15T02:29:35....	0
"2021-06-15T01:36:55....	65.00000000000013	"good"	"HX_heating_inlet-T"	"2021-06-15T02:29:35....	0
"2021-06-15T01:36:55....	54.4548977911067	"good"	"Feed_separator-T"	"2021-06-15T02:29:35....	0
"2021-06-15T01:36:55....	111.201168842527	"good"	"Feed_separator-Press...	"2021-06-15T02:29:35....	0
"2021-06-15T01:36:55....	28491.6594851397	"good"	"Liquid_out-Mass_Flow"	"2021-06-15T02:29:34....	0
"2021-06-15T01:36:55....	25.0000000000001	"good"	"Vapor_out-T"	"2021-06-15T02:29:34....	0

Figure 35. Telemetry Data as input to Stream Analytics without pre-processing applied

One of the possible manners in which it is possible to process the telemetry data, for it to be displayed in graphics depending on its tags, corresponds to the use of User-Defined Aggregates (UDAs) in Stream Analytics. UDAs can be seen as functions defined by the user with the intention of interacting with the telemetry data and transforming it in a specific way. In the case of this project the function has been designed for the data to be transformed, which permits to display it in tables and graphics.

Function alias

FeedPressureUDA

Output type

any

```

1 function main() {
2   this.init = function () {
3     this.state = 0;
4   }
5
6   this.accumulate = function (value, timestamp) {
7     if (value.name == 'Feed-Pressure') {
8       this.state = value.value;
9     }
10  }
11
12  this.computeResult = function () {
13    return this.state;
14  }
15 }
```

Figure 36. Example of an UDA Function

User-Defined Aggregates (UDAs) written in JavaScript have been used so as to pre-process each of the tags received in the telemetry data. In the example from Figure 36, for a given numerical value, if its tag identification matches with 'Feed-Pressure', the numerical value is returned by the Aggregate. However, the use of UDAs has associated a drawback. A 0 is returned if no numerical value related to the tag is read at the timestamp of the execution of the UDA.

Test query

```

1 SELECT
2   timestamp, uda.FeedMassFlowUDA(DataIoTHub) as Feed_Mass_Flow, uda.FeedPressureUDA(DataIoTHub) as Feed_Pr
3   uda.VaporoutTUDA(DataIoTHub) as Vapor_out_T, uda.VaporoutMassFlowUDA(DataIoTHub) as Vaporout_Mass_Flow,
4   uda.VaporoutPressureUDA(DataIoTHub) as Vapor_out_Pressure, uda.FeedseparatorPressureUDA(DataIoTHub) as F
5   uda.TIC100PUDA(DataIoTHub) as TIC_100_OP, uda.TIC100PVUDA(DataIoTHub) as TIC_100_PV, uda.TIC100SPUDA(Da
6 INTO
7   PowerBIVisualizationOutput2
8 FROM
9   DataIoTHub
10 GROUP BY
11   timestamp,
12   TumblingWindow(second,20)
13
14
```

Figure 37. Partial screenshot of the query executed in Stream Analytics

In the query written in SQL, it is possible to see 4 key words (SELECT, INTO, FROM and GROUP BY). Those key words are known as statements, which are to be described briefly below:

1. SELECT. It allows to select specific columns from the telemetry data or to specify the building-up of new columns based on the data read. In this case, timestamp is selected as a column, while a new column is built-up for each different tag received in the telemetry data. Each of these new columns has as heading one of the tags and contains the numerical values associated with this tag.
2. INTO. It is specified the output into which Stream Analytics sends the data. In this case a Table contained in a Dataset from Power BI.
3. FROM. It is specified the input from which Stream Analytics reads the data. In this case a consumer group from Azure IoT Hub.
4. GROUP BY. The criterion under which group the data is indicated. Aggregate functions require to specify this criterion and require a refreshing period, which is of 20 seconds in this case. This refreshing period implies that data is to be read, pre-processed and sent into Power BI each twenty seconds, which permits the synchronization with the rest of modules conforming the communications architecture.

Both the code related to the UDAs and the query executed in Stream Analytics is based on the solutions described at (23).

timestamp	Feed_Mass_Flow	Feed_Pressure	FeedT	Liquidout_T	Liquidout_Mass_Flow	Liquidout_Pressure	Vapor_c
"2021-06-15T01:35:19....	28469.5255325731	390.000000000001	24.5234963397885	54.4374300833336	28467.1419721283	101.300000000001	25.0000
"2021-06-15T01:36:55....	28530.6423565656	390.000000000001	24.5502750386995	0	28491.6594851397	101.300000000001	25.0000
"2021-06-15T01:38:31....	28543.0683230511	389.999999999999	24.6187896331144	54.4405930501997	28475.2713910867	101.299999999999	24.9999

Figure 38. Partial screenshot of the Telemetry Data as input to Power BI after pre-processing is applied

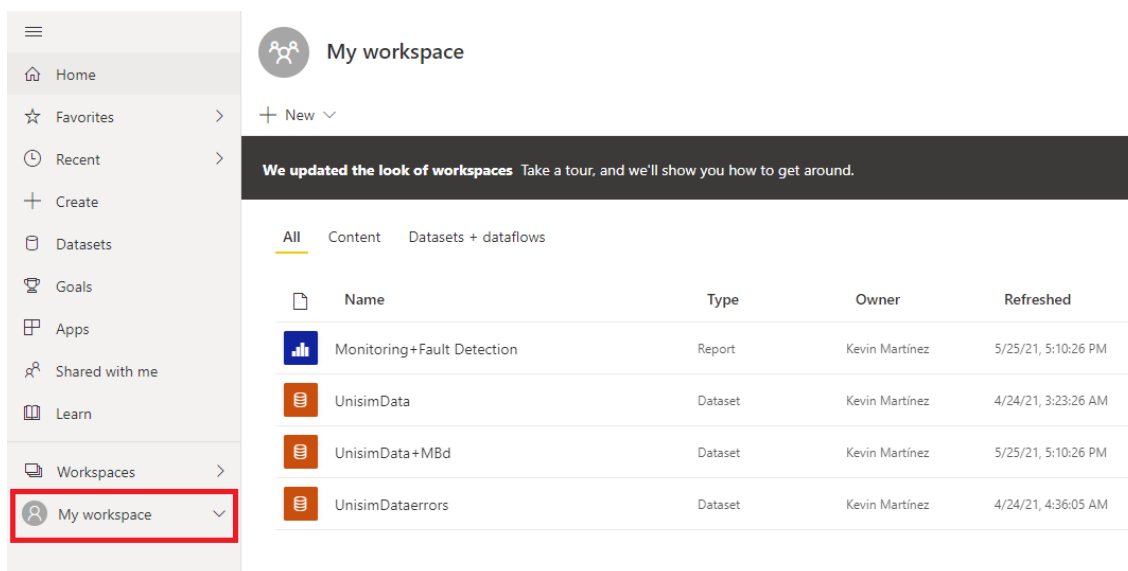
1.11.4. Power BI

Power BI can be described as a solution for entrepreneurial analysis. It permits unifying data from many sources to create interactive, immersive dashboards and reports, in real time, that provide actionable insights and drive business results (24).

The reports and dashboards can be shared among different users belonging to the same organization, which allows them to dispose of the information in real time. The information can be accessed from the computer, the web or the phone since Power BI is available as desktop application, online service (SaaS) and mobile app (25).

In the herein project, a report is created for the display of data from the simulation in the form of graphics and tables. Both of them can be found on the Monitoring Palettes from the report.

The reports and dashboards can be created from a Dataset, which are stored in the My workspace area as it is shown in Figure 39.



Name	Type	Owner	Refreshed
Monitoring+Fault Detection	Report	Kevin Martínez	5/25/21, 5:10:26 PM
UnisimData	Dataset	Kevin Martínez	4/24/21, 3:23:26 AM
UnisimData+MBd	Dataset	Kevin Martínez	5/25/21, 5:10:26 PM
UnisimDataerrors	Dataset	Kevin Martínez	4/24/21, 4:36:05 AM

Figure 39. Example of My workspace area containing Datasets and Reports

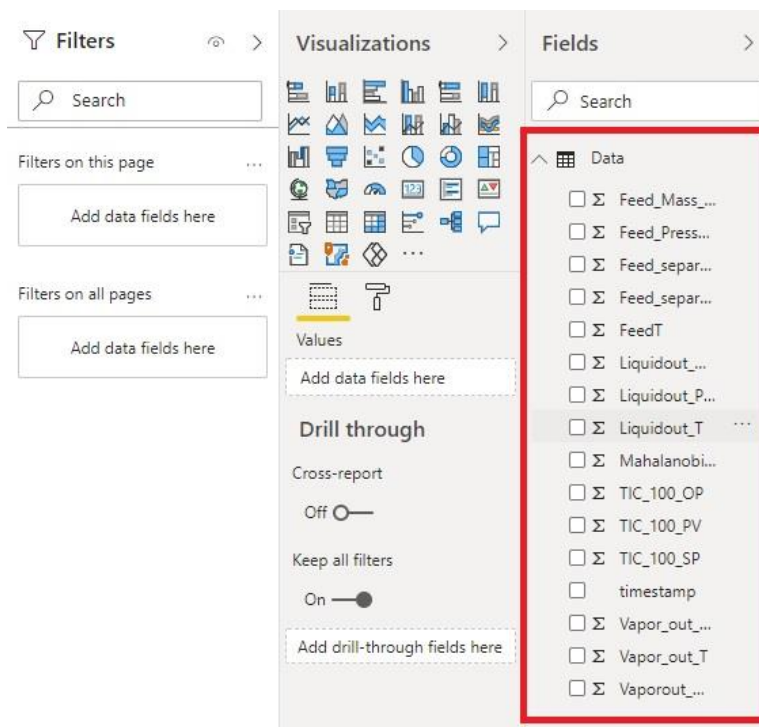


Figure 40. Example of a Table named Data containing different process variables and its corresponding timestamp. Process variables, being numerical, are preceded by a capital sigma, while the timestamp is stored as text.

Prior to having created a report, data is gathered in Power BI in a Dataset. When manipulating a Dataset, there is the option of saving it in the form of a report, which is editable in the future if more data is expected to be received.

1.11.4.1. Monitoring Palettes

A report named *Monitoring + Fault Detection* has been created. The report is composed of different palettes, which are analogous to the sheets from an Excel File. In this section, the monitoring palettes are to be described.

The process variables received from the simulated chemical plant can be monitored in real time with the use of these palettes, which are named Monitoring Table and Monitoring Graphics respectively.

In the Monitoring Table palette that can be seen in Figure 41, all the process variables from the simulation are displayed in a table as a function of the timestamp at which they were received.

Timestamp	Feed_Mass_Flow	Feed_separator_Pressure	Feed_Pressure	Feed_separator_T	FeedT	Ht_heating_inlet	Liquidout_Mass_Flow	Liquidout_Pressure	Liquidout_T	Mahalanobis_distan
2021-06-15T02:22:42.453	27,952.79	111.37	390.00	32.13	26.06	35.00	28,069.29	101.30	33.45	35.7
2021-06-15T02:22:06.495	28,043.39	111.37	390.00	32.16	26.10	35.00	28,156.94	101.30	33.71	59
2021-06-15T02:21:42.513	28,136.42	111.37	390.00	32.16	26.20	35.00	28,236.62	101.30	34.03	31.1
2021-06-15T02:21:06.538	28,216.96	111.37	390.00	32.13	26.13	35.00	28,306.01	101.30	34.42	12.4
2021-06-15T02:19:54.595	28,260.33	111.37	390.00	32.09	26.07	35.00	28,387.75	101.30	34.96	37.1
2021-06-15T02:19:30.618	28,383.02	111.37	390.00	32.11	25.93	35.00	28,444.41	101.30	35.55	13.4
2021-06-15T02:18:54.644	28,476.08	111.37	390.00	32.13	26.07	35.00	28,490.46	101.30	36.23	9.2
2021-06-15T02:18:30.658	28,602.85	111.37	390.00	32.14	26.13	35.00	28,492.70	101.30	37.07	37.1
2021-06-15T02:17:06.731	28,680.95	111.37	390.00	32.13	26.17	35.00	28,475.18	101.30	38.27	17.4
2021-06-15T02:16:42.777	28,697.57	111.37	390.00	32.15	26.12	35.00	28,445.09	101.30	39.51	43.4
2021-06-15T02:16:06.802	28,659.37	111.37	390.00	32.35	26.03	35.00	28,416.19	101.30	41.05	99.4
2021-06-15T02:15:42.812	28,595.07	111.37	390.00	33.31	25.90	35.00	28,402.19	101.30	42.64	28
2021-06-15T02:14:30.872	28,601.87	111.35	0.00	35.55	25.92	37.11	28,397.10	0.00	44.94	96
2021-06-15T02:13:30.900	28,514.38	111.31	390.00	39.88	25.87	43.09	28,420.66	101.30	46.33	109
2021-06-15T02:12:54.910	28,577.20	111.30	390.00	42.07	25.82	46.07	28,419.06	101.30	49.86	41.4
2021-06-15T02:12:30.933	28,566.45	111.28	390.00	44.68	25.79	49.54	28,407.20	101.30	51.45	82.427
2021-06-15T02:11:19.002	28,568.10	111.28	390.00	46.95	25.72	52.49	28,392.10	101.30	52.62	58.016
2021-06-15T02:10:43.020	28,543.43	111.04	390.00	49.30	25.67	55.47	28,376.64	101.30	53.87	86
2021-06-15T02:10:19.052	28,474.77	111.22	390.00	51.54	25.63	58.40	28,366.62	101.30	54.25	65.4
2021-06-15T02:09:43.075	28,439.62	111.21	390.00	53.81	25.50	61.88	28,376.64	101.30	54.65	64.1
2021-06-15T02:08:31.123	28,454.34	111.20	390.00	54.75	25.51	64.73	28,366.58	101.30	54.72	43.168
2021-06-15T02:07:55.167	28,446.47	111.20	390.00	54.79	25.52	65.00	28,361.80	101.30	54.71	60.9
2021-06-15T02:07:31.194	28,398.42	111.20	390.00	54.80	25.63	65.00	28,366.24	101.30	54.70	66
2021-06-15T02:06:55.216	28,316.17	111.20	0.00	54.80	25.52	0.00	28,401.21	0.00	54.67	73.1
2021-06-15T02:05:19.312	28,430.44	111.20	390.00	54.67	25.38	65.00	28,490.14	101.30	54.64	19.4
2021-06-15T02:04:43.327	28,401.70	111.20	390.00	54.69	25.20	65.00	28,521.82	101.30	54.63	83.4
2021-06-15T02:04:19.345	28,395.42	111.20	390.00	54.65	25.21	65.00	28,579.98	101.30	54.62	49.1
2021-06-15T02:04:04.416	28,488.54	111.20	390.00	54.60	25.14	64.00	28,619.06	101.30	54.61	19.4
Total	882,136.59	3,449.84	11,310.00	1,352.19	796.98	1,473.76	881,157.90	2,937.70	1,452.00	187,002.7

Figure 41. Data received from the simulation displayed in tabular form available in the palette Monitoring Table

In the Monitoring Graphics shown in Figure 42, the different process variables are displayed in graphics attending to groups. That is to say, temperatures are displayed in the same graph. The same applies for pressures, mass flows or the data from controllers. However, if some variables' values are too different inside the same group, the variables are displayed separately. The latter applies, for instance, if there are flows ranging from tens to hundreds, while another one is composed of data points with lower values.

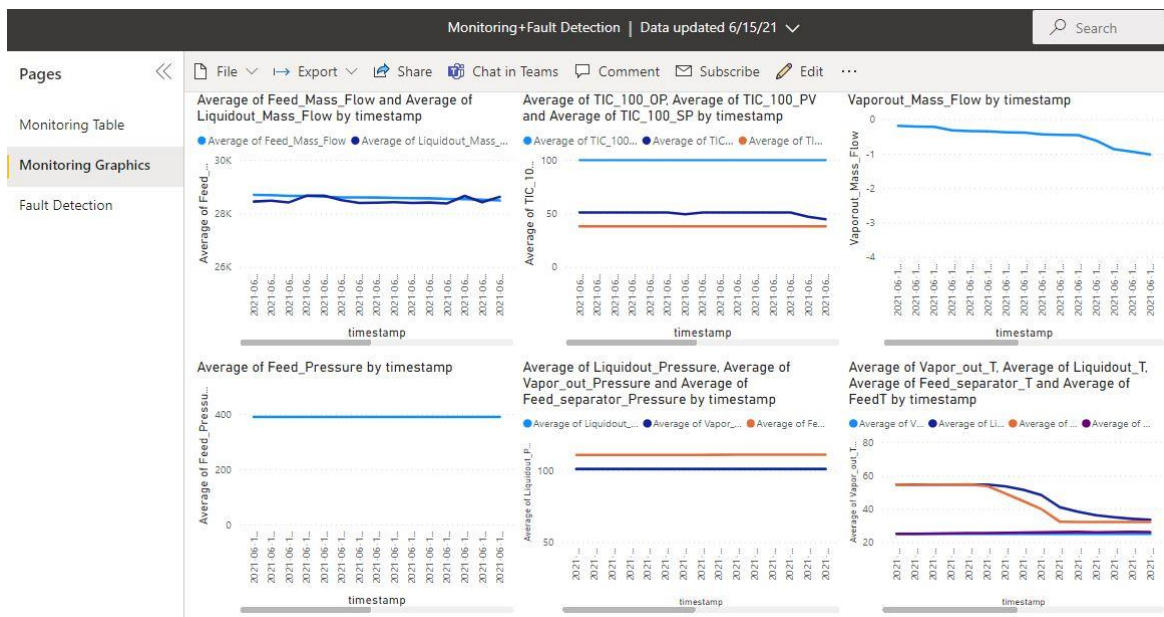


Figure 42. Data received from the simulation displayed in graphics available in the palette Monitoring Graphics

1.11.5. Configuration of the message route

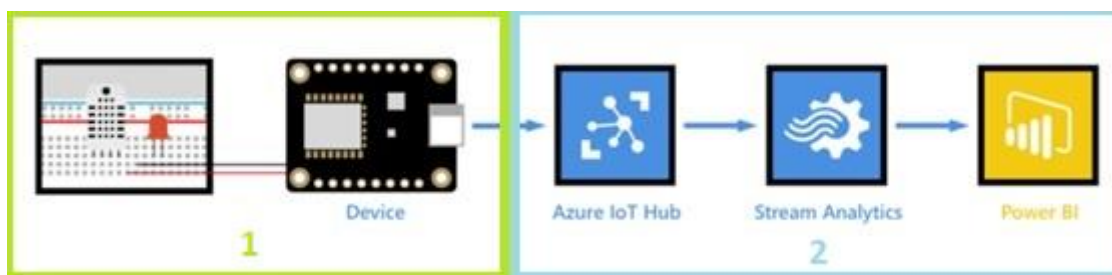


Figure 43. Block Diagram where it is depicted the message route for sending telemetry data from an IoT device to Power BI extracted from (19)

The message route from Figure 43 consists of five blocks: sensors, device, Azure IoT Hub, Stream Analytics, Power BI. The route can be divided in two subunits for its subsequent analysis:

- First subunit. From the sensors to Azure IoT Hub
- Second subunit. From Azure IoT Hub to Power BI

The first sub-unit has already been explained in previous sections of the project. Its blocks are those corresponding to the simulation, the virtual Scada and the IoT device created by the MQTT Broker. The communications employed for routing the messages between the blocks are the ones based on OPC UA and the MQTT Broker. It is possible to state that this subunit can be executed locally, such as from a personal computer.

The second sub-unit is Cloud-hosted as all its constituent blocks are Cloud-based. In this case, the message routing has been configured according to the guidelines provided at (19).

The aforementioned guidelines can be abridged as:

1. To add a Consumer Group to Azure IoT Hub. A Consumer Group can be seen as an end-point where the messages are sent after being received by the device.
2. Add an input to the Stream Analytics Job. To click on Add stream input and select IoT Hub from the drop-down list.

The input feeds from the Consumer Group from Azure IoT Hub. Then, both the Stream Analytics Job and the Azure IoT Hub must have the same consumer group.

3. Add an output to the Stream Analytics Job. To click on Add and select Power BI.
It is necessary to indicate the Dataset Name and the Table Name of the Dataset and Table to which data is sent.

Once configured the output, it is required to authorize its creation by signing-in the desired Power BI account.

1.11.6. Advantages and disadvantages of the Cloud

The incorporation of the Cloud in a communications architecture offers new possibilities that can be translated into advantages. However, as for anything new, advantages and disadvantages come together. In the case of the Cloud, the aforementioned could be summarized as follows:

Advantages:

- Accessibility to the data anywhere and at any time
- Easiness in sharing data
- Infrastructure and electricity costs reduced

Inconveniences:

- Possible failure or unavailability of the Cloud Services
- Dependence on connection to Internet
- Possible exposure to cyberattacks



Development of Machine Learning and Data Analysis algorithms

Algorithms centered on the use of Machine Learning and Data Analysis are to be implemented. On the one hand, the role of the Data Analysis algorithm consists of extracting the main features able to explain the process data generated by the simulated case. On the other hand, the Machine Learning Algorithm is designed so as to feed from these main features in order to generate an indicator, which could be used as a red flag for knowing the status of the chemical plant being simulated. In other words, it is intended to have an indicator, which could detect whether there are faults or abnormalities occurring in the chemical plant.

Both algorithms are to be executed on-line together with the communications algorithm and the modular communications architecture in order to upload this indicator to the Cloud Platform for its display together with the process variables from the simulated case, which are to be monitored on-line.

1.12. Theoretical background

1.12.1. Normalization of the data and pretreatment steps

1.12.1.1. Normalization of the data

It is intended to perform a Principal Component Analysis (PCA) over the obtained dataset. This method is based on the identification of those directions in which the variance of the data is the greatest. The variance from a variable is measured in its scale squared. If prior to performing the method, the variables are not standardized for them to have a mean of 0 and a standard deviation equal to 1, those variables with greater scale will dominate over the rest. In consequence, the standardization of the dataset is required prior to performing a PCA (26).

1.12.1.2. Pretreatment steps

Depending on the nature of the dataset, namely, how it is generated or how data is transferred upon receipt, it is possible the dataset to be contaminated. For instance, the dataset can contain:

- Nan values. This could be related to datasets where certain variables could not always be measured. For example, a dataset containing variables describing information relative to people. In this case, people are not forced to provide all the information required, which could

be the cause for certain variables to be unknown. Another simpler explanation could be that certain variables cannot always be measured. Nan values could be substituted for the variable's mean or mode, or just be eliminated if the dataset is larger enough.

- Abnormal values that deviate from the mean value. An assessment relative to how treat these values should be performed. An option could be to eliminate those abnormal values in a filtering stage.
- No presence of values for a given variable. For example, a dataset obtained from the Cloud or OPC UA Communications could contain 0s symbolizing that, for a given timestamp and variable, no data was received. An option could be a filtering stage consisting of the substitution of the 0. It could be substituted by the directly previous value adopted by the variable affected by the communications gaps.

1.12.2. PCA Method

All the information relative to the PCA method is based on the explanations available at (26).

1.12.2.1. Objective of the method

Principal Component Analysis, commonly abbreviated as PCA, is a statistical method that allows to simplify a complex sample space accounting for a great number of dimensions, by reducing its dimensions, while keeping the information it contains.

If having a sample of n records, each one of them being explained by p variables (X_1, X_2, \dots, X_p). In other words, the sample space has p dimensions. The use of PCA permits finding a number of subjacent factors ($z < p$), which explain the same as the original p variables approximately. That is to say, after applying the PCA to a sample space, each record can now be explained by z variables, instead of p variables. Each one of these z variables is known as principal a component.

1.12.2.2. Calculation of each principal component

Each principal component (Z_i) is obtained through the linear combination of the original p variables (X_1, X_2, \dots, X_p). For instance, the first principal component (Z_1) could be mathematically expressed as the normalized linear combination of the p variables:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p \quad (1)$$

ϕ_{j1} : loading for j variable

X_j : variable j from the set of p variables

Z_1 : first principal component

Since the linear combination is normalized, it is fulfilled that:

$$\sum_{j=1}^p \phi_{j1}^2 = 1 \quad (2)$$

ϕ_{j1} : loading for j variable

The different loadings ($\phi_{11}, \dots, \phi_{p1}$) can be interpreted as the weight or importance that each original variable has for each component. This weight is related to the kind of information picked up for the component. The values of these weights are optimized for maximizing the variance expressed by the principal component. The amount of variance expressed by a principal component is linked to the importance of that component. The greater the variance, the more important the component is. The components are ordered as a function of the variance they express, hence, Z_1 accounts for the maximum variance, Z_2 expresses minor variance than Z_1 , ..., up to Z_n that expresses the littlest variance.

Once the first principal component (Z_1), expressing maximum variance, is calculated, Z_2 is determined by adjusting its corresponding weights. However, the condition of no correlation with Z_1 and the linear combination associated with Z_2 is imposed. That is equivalent to impose Z_1 and Z_2 to be perpendicular. Each subsequent component is calculated afterwards by imposing perpendicularity between the given component and the rest of components.

1.12.2.3. Number of principal components to calculate

Given a dataset composed of n data points and p components, the number of principal components up to which it is possible to reduce the dataset dimensions are $n-1$ or p , whichever is the limiting factor. However, the objective of the method is related to reducing the dimensions of the dataset. Thence, the number of principal components is usually associated with the minimum number of p variables that allows to retain the optimal amount of information relative to the original dataset.

For determining that minimum number of p variables, the number of components to retain is usually established according to the amount of variance that it is desired for the new sample space to express.

It is usually wanted to obtain a sample space that minimizes the loss of information, but which also permits to work with a minor dimensional space than the original dataset. For determining the amount of variance associated with each principal component, the variance proportion and the explained accumulated variance proportion are the values that are usually studied.

For a given dataset, all the principal components are determined. The new dataset obtained is a transformation based on the linear combination of the original dataset. However, all the information, although transformed, is being preserved without loss of information. The summation of the explained accumulated variance proportion associated with this transformed dataset, with principal components equal to the number of variables from the original dataset, is equal to 1. There is no loss of information. Each principal component has a variance proportion associated, the sum of which corresponds to 1.

Therefore, the number of principal components to retain corresponds to the number by which there is not a substantial increase on the explained accumulated variance proportion. For instance, the graph below shows the explained accumulated variance proportion as a function of the number of principal components picked up.

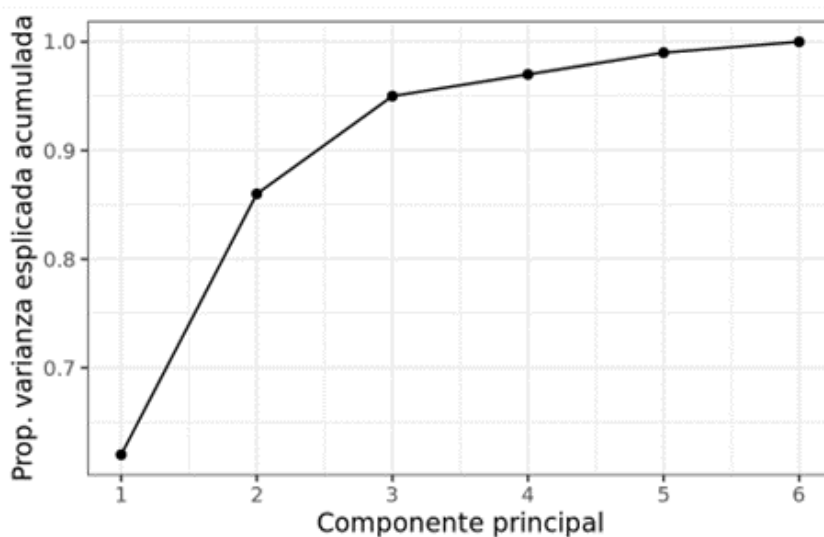


Figure 44. Explained accumulated variance proportion – Principal Component Graph

From the graph, it is possible to conclude that the first principal components preserve important amounts of variance, but subsequent principal components tend to pick up less and less variance. If picked up 4 principal components, more than 95% of the variance from the original dataset is preserved. If it is possible to lose 5% of the variance, which corresponds to minimum loss of information, then 4 will be the number of principal components so as to calculate.

1.12.3. Mahalanobis distance

The Mahalanobis distance (MD) is the distance between two points in multivariate space. In a regular Euclidean space, variables (e.g., x , y , z) are represented by axes drawn at right angles to each other. The distance between any two points can be measured with a ruler. For uncorrelated variables, the Euclidean distance equals the MD. Nevertheless, if two or more variables are correlated, they cannot be represented in axes at right angles or be measured by hand. Moreover, if the number of dimensions is superior to three, the data cannot be plotted in an understandable manner.

The Mahalanobis distance permits to measure distances between points, which can be correlated and of a great number of dimensions. It is based on the measurement of the distance relative to the centroid, which can be interpreted as an overall mean for multivariate data. The centroid is a point in multivariate space where all means from all variables intersect. The larger the MD, the further away from the centroid the datapoint is (27).

This distance permits to identify if a datapoint is abnormal since it can be detected as being further away from the centroid. That is to say, the datapoint has associated atypical values with respect to the mean values, accordingly being considered an outlier. Thereby, MD can be applied for the identification of outliers present in a dataset.

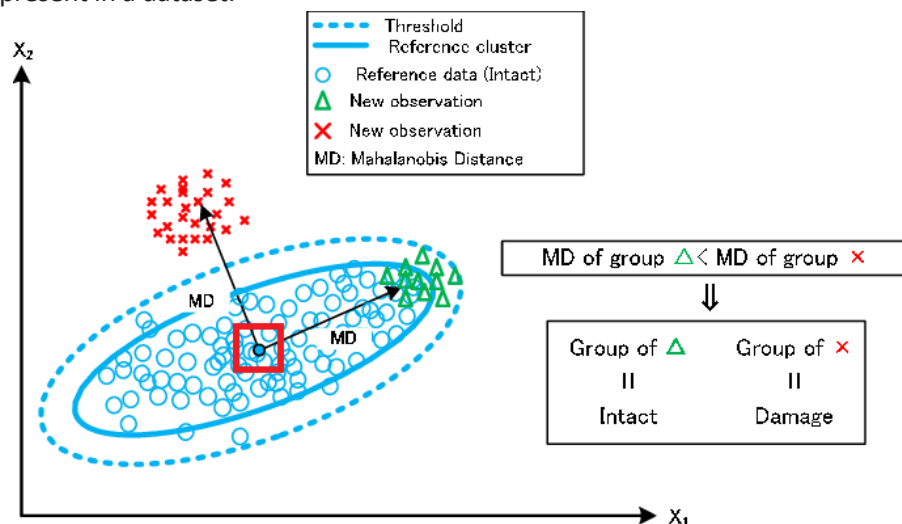


Figure 45. Mahalanobis Distance for its application on the detection of outliers extracted from (28)

For example, if the multivariate space is depicted in a two-dimensional Euclidean space and accounting with non-correlated variables as it is seen in Figure 45, the centroid corresponds to the blue dot delimited by the red square. On the one hand, the Mahalanobis Distance of the group of data points in green is considered to be normal and the data points are labelled as “Intact”. On the other hand, the Mahalanobis Distance of the group of data points in red is abnormally high and this group is labelled as outliers or “Damage”.

1.13. Python code functions for communications combined with machine learning and data analysis

1.13.1. Implementation of machine learning and data analysis functions

The functions that are to be commented along this section are based on the explanations and code available at (29) and (30) respectively. The functions are intended for getting an algorithm able to:

- Perform tasks of data analysis with the data extracted from the simulated case in UniSim.
- Generate predictions by means of machine learning using the data obtained through data analysis.
- To send on-line predictions to the Cloud Platform for its display in graphical form in Power BI. That is to send predictions at the same time the simulated case is running. For that, the same communications interface developed for monitoring is to be used. In fact, both the predictions and the data to be monitored are to be sent simultaneously.

The same methodology to explain, as the one used for Python functions employed for communications and monitoring, is to be followed. Thus, the functions are to be explained conceptually according to the missions and objectives that must be achieved. It is possible that for a given objective or mission, several functions are to be described as a group.

Dataset. It consists of a function that permits to store the data from the process variables listed in the Process Data Table *ProcData1*. The data is saved in a spreadsheet of an Excel file, following the format of assigning the numerical values from each variable to a specific column from the spreadsheet. The dataset used in this project consists of 5700 data points approximately.

The execution of the **Dataset** function occurs in a loop that is repeated automatically as in the same way as with the **Store Data** function. The function stops when the User pushes the “Stop Integrator” button.

PCA Method. This function permits to perform data analysis tasks both for the dataset and for the data read on-line. Regarding the dataset, it loads the data saved in the Excel file, normalizes it and performs the PCA method over it. Regarding the data read on-line, the steps are the same but the data is processed in groups of sixteen data points. It is necessary to remember that the number of data points to which the method is to be applied must be always greater than the number of principal components, a condition that is accomplished for both the dataset and the data read-online. In consequence, the amount of on-line data points permits the use of the PCA method and to perform quick data readings for on-line processing.

Relative to how the PCA method has been implemented, an aspect must be commented on. Specifically, the one related to the variance captured by the principal components to which the original dataset dimensions are reduced.

Considering the reduction of the dataset dimensions to three principal components, it has been studied the variance captured by the principal components analysis when data is read on-line. Depending if abnormalities are provoked to occur in the simulation case or not, the variance captured by the principal components analysis changes.

```
variance PCA per component_____
[0.88001016 0.07804698 0.0266764 ]
```

Figure 46. In descending order of weight and normalized to 1, variances captured by each of the three principal components to which the dataset has been reduced before abnormalities are provoked. Results obtained in Python.

```
variance PCA per component_____
[9.78712118e-01 2.00218544e-02 9.08849314e-04]
```

Figure 47. In descending order of weight and normalized to 1, variances captured by each of the three principal components to which the dataset has been reduced after applying a temperature ramp of $-4^{\circ}\text{C}/\text{minute}$ on the stream HX_heating_inlet. Results obtained in Python.

It is possible to infer that prior to the presence of abnormalities, the principal components are able to capture more than the 98% of the variance within the original dataset. However, this variance captured by the components gets inconsiderable after abnormalities are provoked.

Since the use of five principal components permits capturing more than 98% when the chemical plant is being simulated at nominal conditions, it has been established to use five principal components in the PCA method. What is more, how the variance captured changes as a function of the presence or not of abnormalities could be used as a complementary indicator together with the Mahalanobis distance. Nonetheless, the stability of the code has been proved not to be always good when this variance captured is calculated together with the data reading and the determination of the Mahalanobis distance. Thus, the Mahalanobis distance is selected as the preferred indicator as its stability is major.

When data is read on-line, it is generated graphics displaying the data points as a function of the different principal components. The graphics can be seen in the Plots Pane from Spyder and also can be saved in the computer. In the graphics, it is possible to appreciate changes relative to the position of the data points relative to the principal components, but those have been used as an extra aid not as the main indicator for the presence of faults.

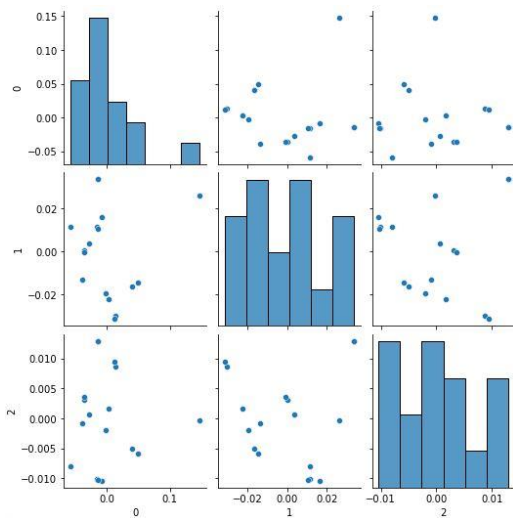


Figure 48. Graphics depicting the data points as a function of the principal components when the plant is at nominal conditions.

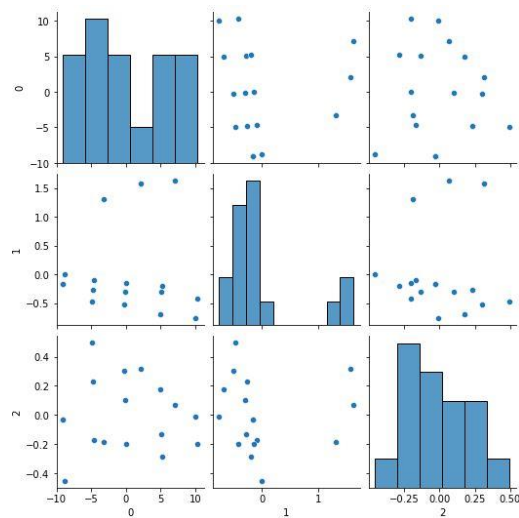


Figure 49. Graphics depicting the data points as a function of the principal components after abnormalities are provoked.

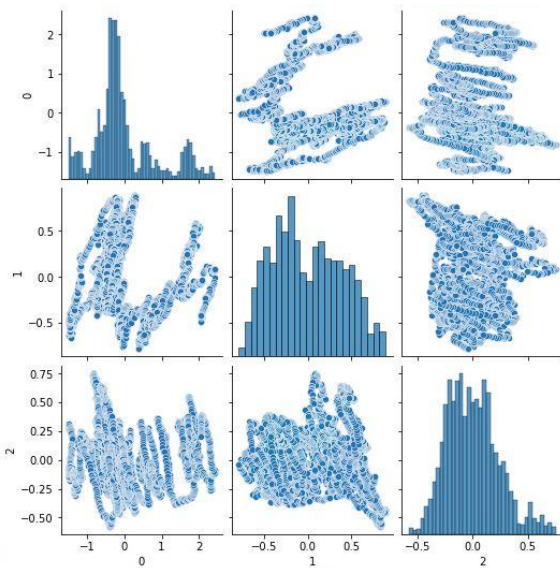


Figure 50. Graphics depicting the data points as a function of the principal components for the dataset

After visualizing Figures 48 and 49, it is concluded that there is an increase in the values for the first principal component identified in both images as 0 in the axis of the graphics. At nominal conditions, the values range from -0.05 to 0.15, while after faults are introduced in the simulated case, the values range from -10 to 10. Other changes on the ranges can also be appreciated on the rest of principal components.

Dist. Mahalanobis. It permits to calculate the Mahalanobis distance for all the data points from the dataset saved in the Excel file and establish a threshold value for the Mahalanobis distance above which a datapoint can be considered as an outlier.

After the calculation of the threshold value, the algorithm is iterated so as to perform filtering stages aimed at eliminating the values considered as outliers. After each iteration, a new threshold value is determined up to an iteration at which no more outliers are detected. At this given iteration, the threshold value is known as threshold without outliers, while the first threshold value prior to any filtering stage is known as threshold with outliers.

```
mb threshold without outliers:  
13.149146482665902  
mb threshold with outliers:  
44.329817418145076
```

Figure 51. Threshold values for the Mahalanobis Distance. Results obtained in Python.

In consequence, the algorithm works as a machine learning algorithm able to give predictions over if a datapoint can be considered normal or outlier. The data with reduced dimensions, obtained after applying the PCA methodology, is used as the input data for the algorithm.

Apart from using the function so as to extract data from the dataset, it can also be used so as to calculate the Mahalanobis distance for data read on-line while the simulated case is running. In this case, the data does not suffer filtering stages since it is desired to preserve all the original values either to save them or to send them to the Cloud Platform. Then, the threshold value, which is valid for online-data, corresponds to the threshold with outliers.

Store Data. This set of functions was previously explained in the sections devoted to communications and monitoring, but it was explained only from that point of view. The part linked to Data Analysis and Machine Learning was not taken under consideration in those sections since the required concepts for proper understanding had not been introduced at that point.

However, the set of functions does include the functions related to Data Analysis and Machine Learning, while it also consists of the functions associated with communications and data monitoring. The set works as follows:

- Reading of sixteen data points on-line.
- The data points are normalized and the PCA method is performed. For normalizing the online-data, the scaler employed is the same as the one that normalizes the dataset. It is important to have the online-data normalized in the same way as the dataset, for the threshold obtained with the machine algorithm to be valid over the online-data.

- The calculation of the Mahalanobis distance for each of the sixteen data points.
- Obtention of the top and the mean values of the Mahalanobis distance from the sixteen values of the Mahalanobis distance, and its sending to the Cloud Platform with the use of the OPC UA Client and the rest of the modular communications architecture.

Apart from being sent to the Cloud, the sixteen Mahalanobis distances calculated are also saved in an Excel file, where each batch of data points is saved in a column from the spreadsheet.

1.13.2. Built-in interface (GUI library Tk)

After having introduced and explained the functions related to Data Analysis and Machine Learning, the complete Built-in interface with all the buttons can be observed in Figure 52.

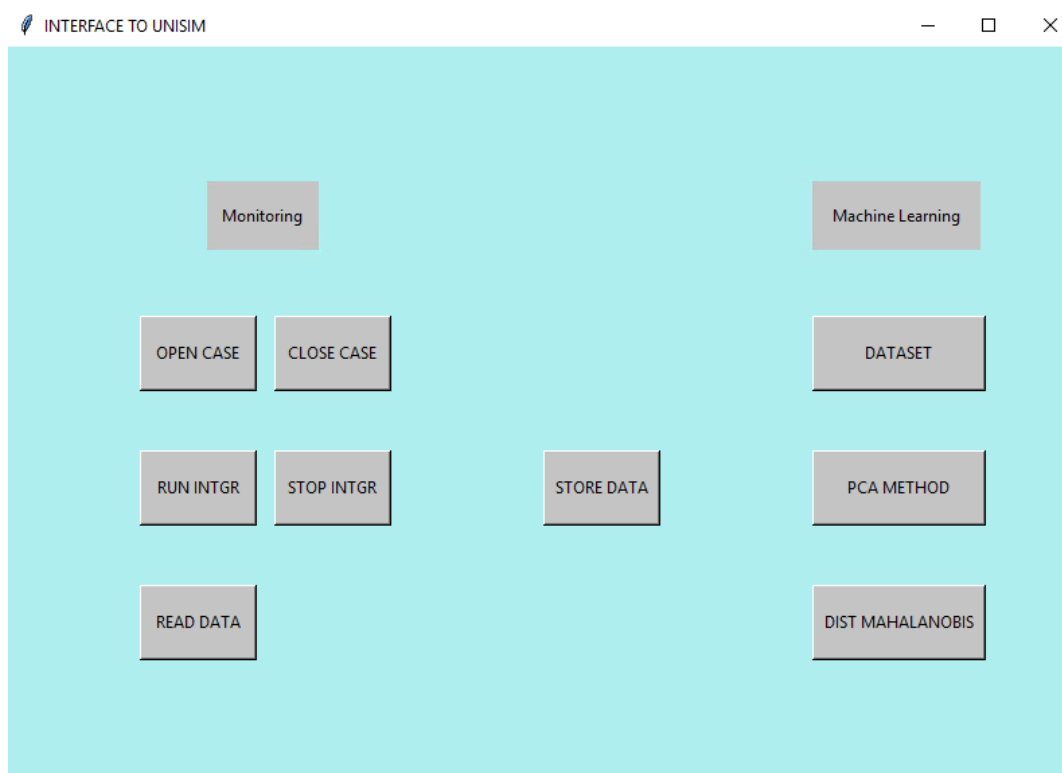


Figure 52. Graphical User Interface for Monitoring, Data Analysis and Machine Learning, and Communications

1.14. Sequence Diagram for communications, machine learning and data analysis in UML standard

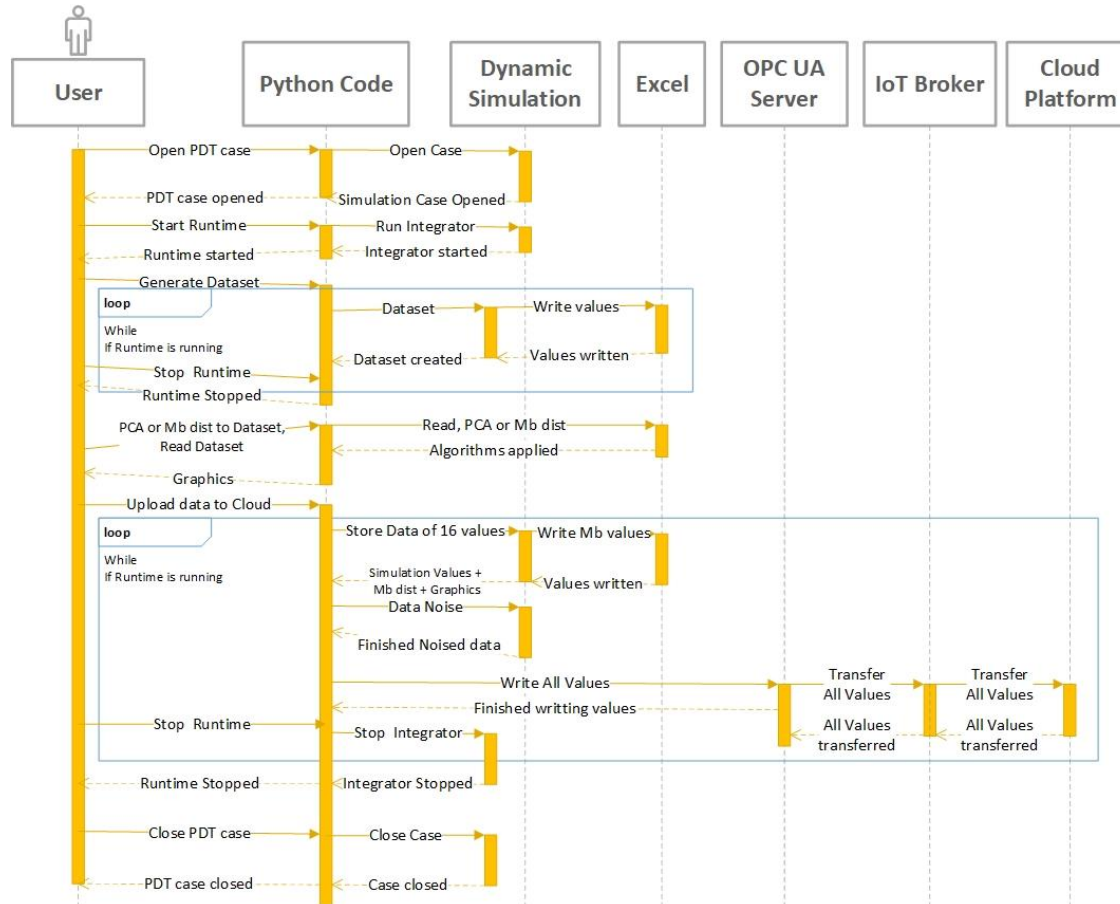


Figure 53. Sequence Diagram for communications, machine learning and data analysis in UML

1.15. Other potential applications of Cloud computing tools

1.15.1. Current potential related to the use of the Cloud

As it is possible to conclude from section 1.11 the use of the Cloud broadens the possibilities related to the communication, storage and display of Data. Moreover, it also permits new ways in which to treat and study the data. For instance, some interesting services offered by the Microsoft Azure Cloud are:

- Microsoft Azure Machine Learning Studio. It enables the use of a Machine Learning Algorithm (MLA) stored in the Cloud.
Through the use of this service, it is also possible to interact with the MLA hosted in the Cloud with Python or other programming tools. It is possible to ask for the retrieval of results calculated by the MLA given an input of data sent to it via the communications protocols offered by Microsoft and compatible with Python and other programming tools.
- Azure Time Series Insight. It permits to analyze temporal series of data

On the one hand, other services available at the Cloud are those related to the storage of data. These services also permit the option of reading the data contained in storage services by using, for example, Python. On the other hand, there are also services to post the data in webs for its display to third-parties.

Development of monitoring tools for the detection of abnormal events in a simulated plant

On the one hand, this section is devoted to discuss how data analysis and machine learning algorithms should read the data of the simulated case. On the other hand, once the algorithms read data, it is described the optimization process followed so as to improve the consistency of the results obtained and get functional algorithms. Once the algorithms are optimized, they are combined with the modular prototype for the exchange of information in order to obtain monitoring tools for the detection of abnormal events.

1.16. Reading of data by the data analysis and machine learning algorithms

As it has been explained in section 1.13.1, the code developed makes possible to generate a dataset, extract information from it, and read data on-line, which is to be compared with the dataset for detecting the presence or not of differences between the dataset and the data on-line. Once the capabilities of the code are known, it is necessary to establish what kind of data can be read as the dataset and which kind can be read as on-line data, with the purpose of getting a functional machine learning algorithm.

The first point to clarify is related to how many kinds of data are generated during the activity of a chemical plant and how to differentiate them.

In Figure 54, it is possible to see the lecture of one of the variables of the simulation. In this case, the variable corresponds to a temperature. For the different variables that are simulated, similar trends could be observed. The key idea to bear in mind is that for the activity of the plant along the year or any temporal period that it is chosen for analysis, it is possible to see what it is shown in the graph below. Concretely, three different populations of Data can be distinguished: A, B and C. If analyzed, on the one hand, A and C correspond to populations of data at nominal conditions but at different set-points. On the other hand, population B corresponds to data associated with a transition between set-points A and C. In conclusion, there are two main kinds of data, which are possible to distinguish depending if they are generated during nominal conditions, or during transitional regime.

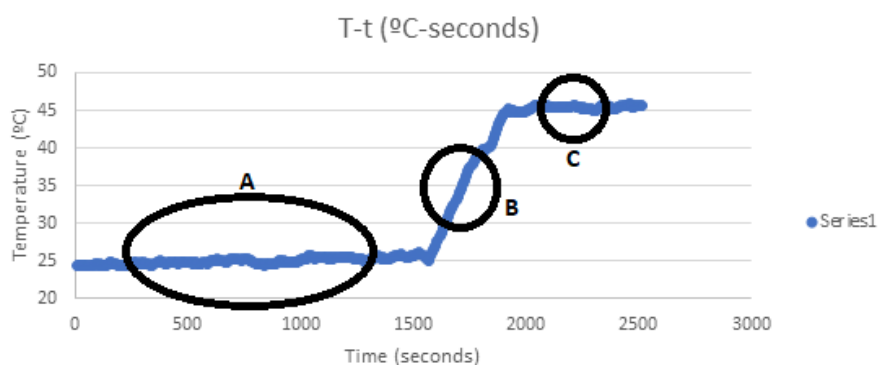


Figure 54. Different data populations during the evolution of temperature as a function of time

It is decided that the dataset will be generated with data obtained when the plant simulated is at nominal conditions. That is when it is at set-points A or C of Figure 54. Accordingly, the machine learning algorithm will learn from data at nominal conditions and will establish a threshold value for the Mahalanobis distance above which the plant is considered not to be at nominal conditions. On the other hand, the on-line data read could be related to nominal conditions of the plant or to a transitional regime experimented by the plant.

Assuming that the plant stays at nominal conditions and no change on the set-points is ordered, the presence of transitional regimes should be associated with abnormalities or faults occurring at the plant. In consequence, depending if the on-line data corresponds to nominal conditions or to transitional regimes experimented by the plant, the Mahalanobis distance calculated is suspected to vary. The Mahalanobis distance will be monitored and if it surpasses the threshold value indicated by the machine learning algorithm, it will be indicative for the plant being in a transitional regime as a consequence of the presence of abnormalities or faults in the plant.

1.17. Preparation of simulation cases for training and testing the machine learning algorithm

In this subsection, it is explained how the simulation cases have been prepared depending if the machine learning algorithm is to be trained or tested.

1.17.1. Simulation case for training the machine learning algorithm

The simulation case prepared for the training of the machine learning algorithm consists of letting the simulated case run without provoking any fault within the plant. Some of the process variables suffer the noising effect described in section 1.9.2.2 in order to record realistic process data, which could

resemble the one that could be recorded in an industrial facility. The process data is stored in an Excel file for its subsequent use as a dataset for the training of the machine learning algorithm.

1.17.2. Simulation case for testing the machine learning algorithm

The simulation case prepared for the testing of the machine learning algorithm consists of submitting the simulated case to a particular fault while running. The fault is introduced in the simulated case in the form of a temperature ramp generated by the Transfer Function shown in Figure 55. A temperature ramp can be defined as the variation experimented by a temperature during a given period of time. Additionally, the simulated case is also affected by the noising effect.

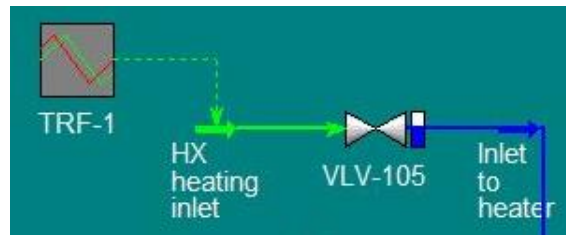


Figure 55. Transfer function used so as to introduce faults in the simulated case

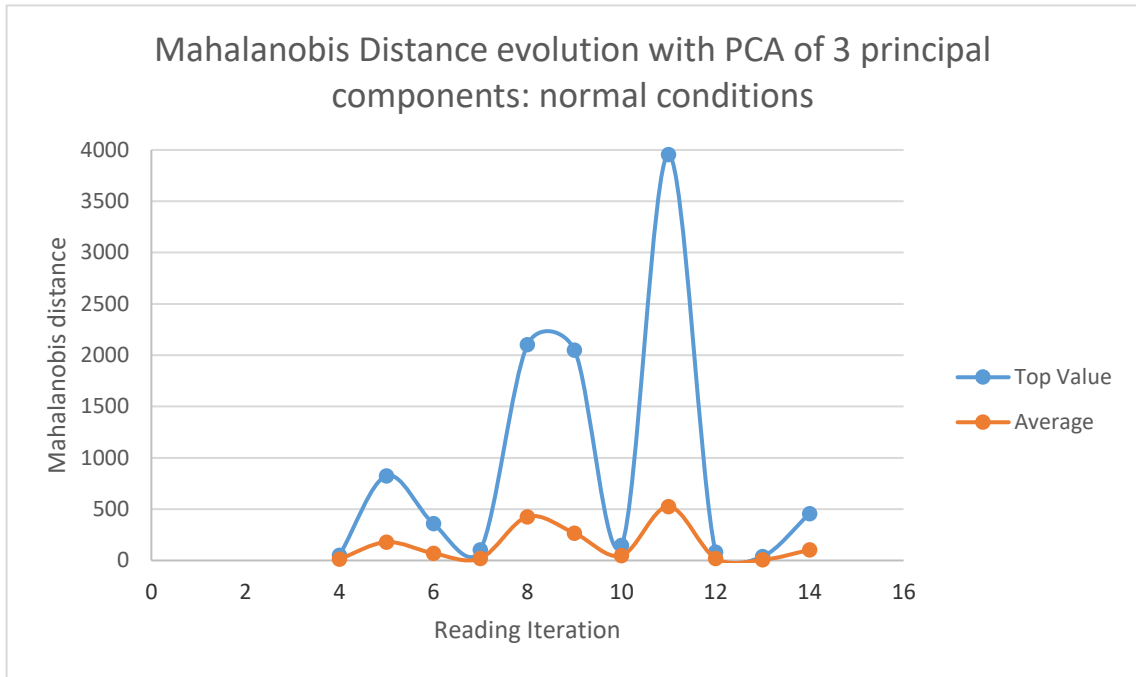
The data recorded is processed by the trained machine learning algorithm and the predictions produced by the algorithm are sent both to the Cloud and to an Excel file.

1.18. Optimization of the Data Analysis and the Machine Learning Algorithms

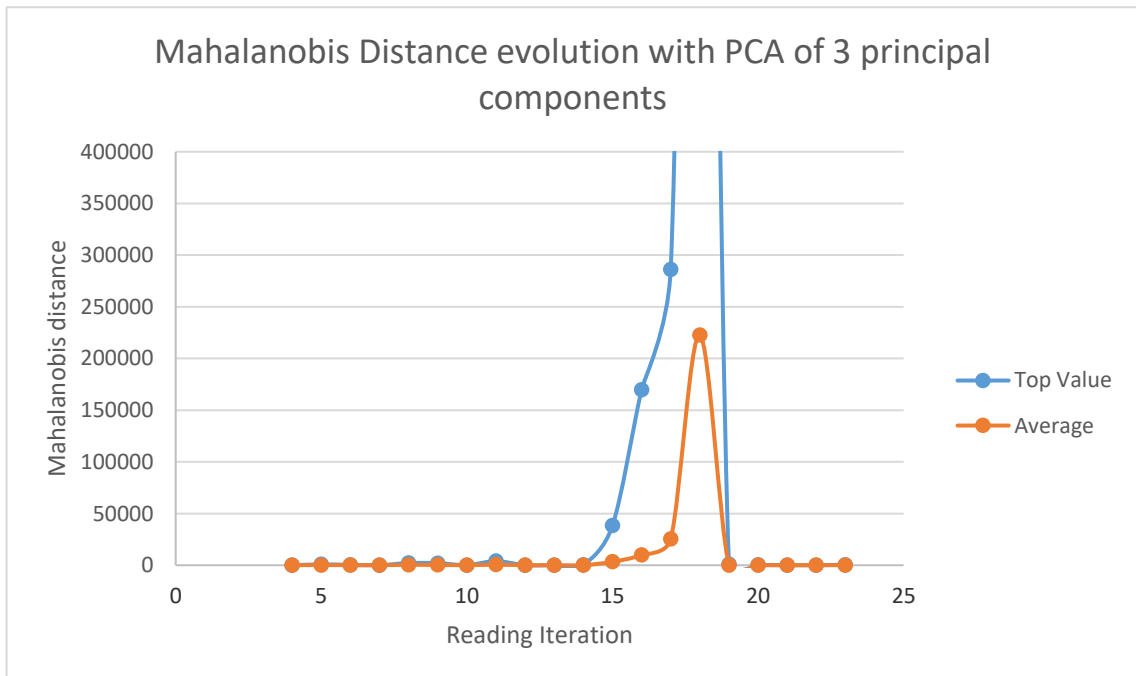
It has been verified with the Python code that depending on the number of principal components for the PCA method, the volatility associated with the variable *Mahalanobis distance* (candidate to indicator) varies. Thus, it has been adjusted the number of principal components so as to minimize the loss of information from the dataset and also to reduce the volatility associated with the Mahalanobis distance.

For testing and assessing the impact of the number of principal components on the evolution of the Mahalanobis distance, the on-line readings have been saved in an Excel file. An Excel file has been generated for each number of principal components, denoted as N , from now on. In this project, it has been studied the cases $N=3$ (Figure 56), $N=5$ (Figure 57).

Case N = 3

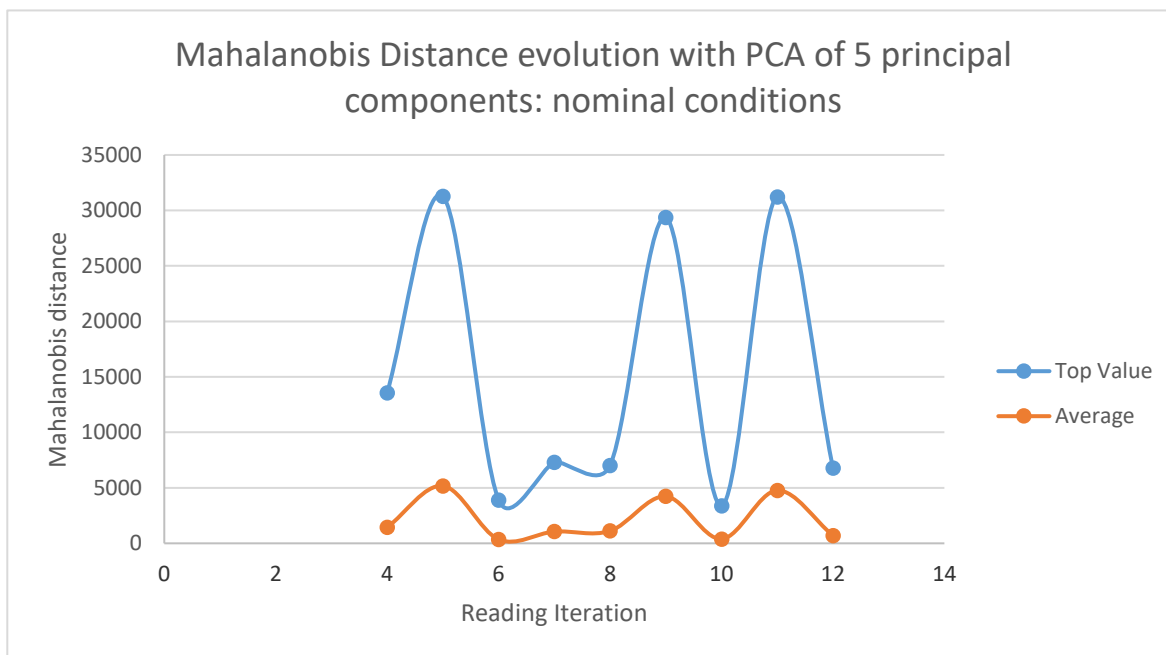


Graph 1. Mahalanobis Distance evolution with PCA of 3 principal components: normal conditions. The average value of the indicator remains at low values similar to the threshold with outliers' value, obtained by the Machine Learning Algorithm after the inspection of the dataset. This permits the use of the threshold value. Additionally, the sensitivity of the top value is greater than the sensitivity of the average value.

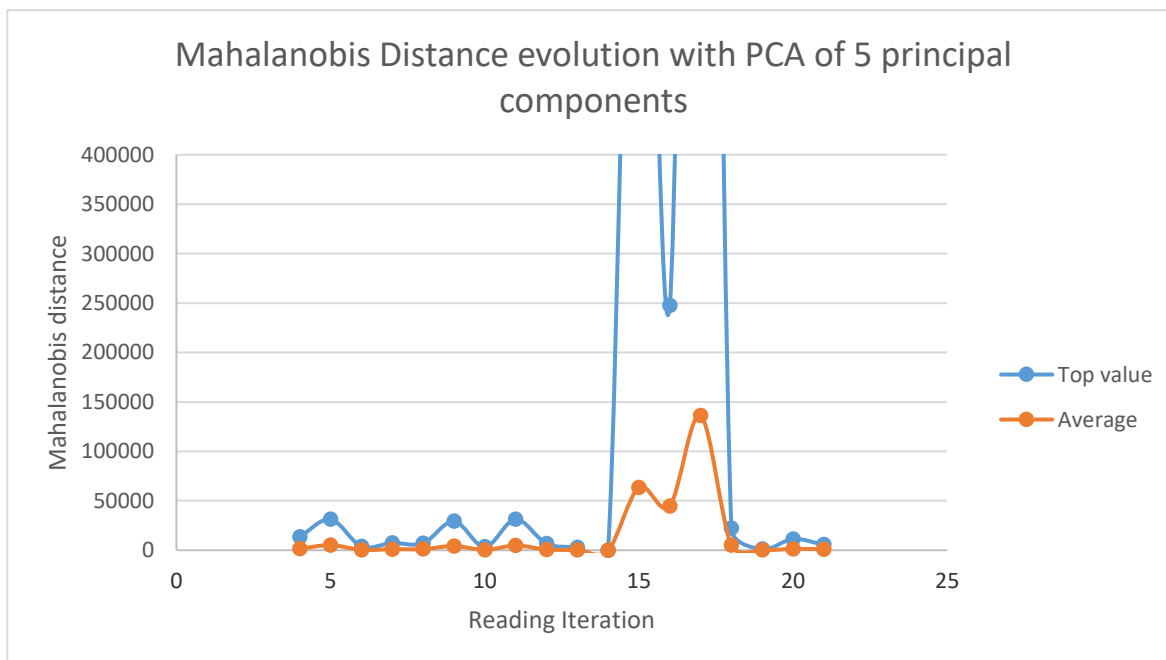


Graph 2. Mahalanobis Distance evolution with PCA of 3 principal components. After the introduction of the fault, the indicator's value evolves exponentially reading after reading. However, the indicator's value decreases to values prior to the fault after the transient period is finished, in other words, after the Temperature ramp stops acting over the stream HX_HEATING_INLET.

Case N = 5



Graph 3. Mahalanobis Distance evolution with PCA of 5 principal components: nominal conditions. The average value of the indicator oscillates from low values to high values due to the high volatility experimented by the indicator. This behavior is detrimental since it is not possible to use the threshold with outliers' value, obtained by the Machine Learning Algorithm after the inspection of the dataset, as a threshold.



Graph 4. Mahalanobis Distance evolution with PCA of 5 principal components. After the introduction of the fault, the indicator's value evolves exponentially reading after reading. However, the indicator's value decreases to values prior to the fault after the transient period is finished, in other words, after the Temperature ramp stops acting over the stream HX_HEATING_INLET.

After the inspection of the graphics for each case, two main conclusions are extracted.

In the first place, it is proved that with the use of a machine learning algorithm, it is possible to identify the presence of a fault within the plant independently of the volatility associated with the indicator. However, the indicator stops working after the transient regime, associated with the introduction of a fault, ends. This is a drawback of the actual algorithm since it identifies a fault in the plant while there is a transient regime. The idea of linking the presence of a transient regime to the presence of a fault is not always valid. For instance,

- A transient regime can occur in the plant due to a change on a set-point by the operators. If a set-point related to a process variable is changed, there is no fault in the plant, but an intentional change over the plant by the operators.
- The presence of a fault in a plant is not always linked to the existence of a transient regime. The transient regime lasts in the plant from the generation of the fault up to the stabilization of the plant. All the same, the fault still exists within the plant after the transient regime ends and the plant arrives at a new steady-state.

The two previous facts imply that linking the presence of a transient regime to the presence of a fault can serve, and serves for the identification of faults. Nonetheless, this identification is subject to limitations. It is only valid when there are no changes on the set-points by the operators, and only serves to identify the generation of the fault, during the transient regime. The indicator deactivates after the plant stabilizes to a new steady-state. Bearing in mind these constrictions, and acting in consequence, the use of the algorithm is valid for fault diagnosis.

In the second place, it is concluded that for $N=3$, the volatility associated with the indicator is far less than the one linked to the indicator when $N=5$. Moreover, the high volatility for $N=5$ is detrimental since the use of the threshold value renders useless being that the indicator keeps oscillating along the time around the threshold value.

To sum up, the best candidate for further development is the algorithm with $N=3$, with which the threshold value could be used in the form of an alarm in more advanced setups of the monitoring tools in future projects.

An additional conclusion can be figured out. The sensitivity of the top value is greater than the sensitivity of the average value. Thus, the monitoring of the top value can be complementary to the monitoring of the mean value due to it makes it possible to identify with major sensitivity the presence of faults.

1.19. Fault Detection Palette

The Machine Learning and the Data Analysis algorithms are combined with the modular prototype for the exchange of information in order to obtain monitoring tools for the detection of abnormal events.

Then, the variables of the mean and top values related to the Mahalanobis distance calculated for each batch of sixteen data points are sent via the modular communications architecture to the Cloud Platform together with the process variables of the simulated case.

These two variables are graphed in the Fault detection Palette, available at Power BI, together with the evolution of the HX heating inlet's temperature. The latter is the variable to which abnormalities are provoked in the form of a negative temperature ramp.

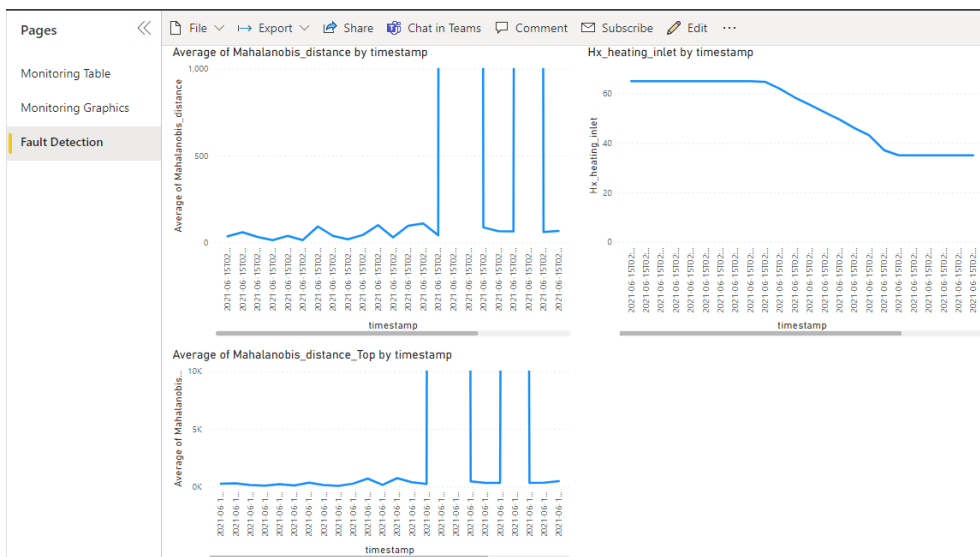


Figure 58. Data received from the simulation displayed in graphics available in the palette Fault Detection

In this palette it is possible to visualize the evolution of the Mahalanobis distance, which acts as an indicator. Once a predefined threshold value for the indicator is surpassed, it is feasible to state that the simulated plant is suffering abnormalities.



Validation of the prototype and guidelines for future development

1.20. Validation of the prototype

1.20.1. Proof of concept

The proof of concept consists of the execution of the simulated case and the on-line monitorization of the indicator known as the Mahalanobis distance, both in the form of its mean and top values, together with the process variable to which the fault has been introduced.

The monitorization is performed both from the Fault Detection Palette available at Power BI and from the Datalogger in UniSim, where several temperatures are displayed. Looking at the figure below, it is proved that while the simulation is running, the data is being received in Power BI.

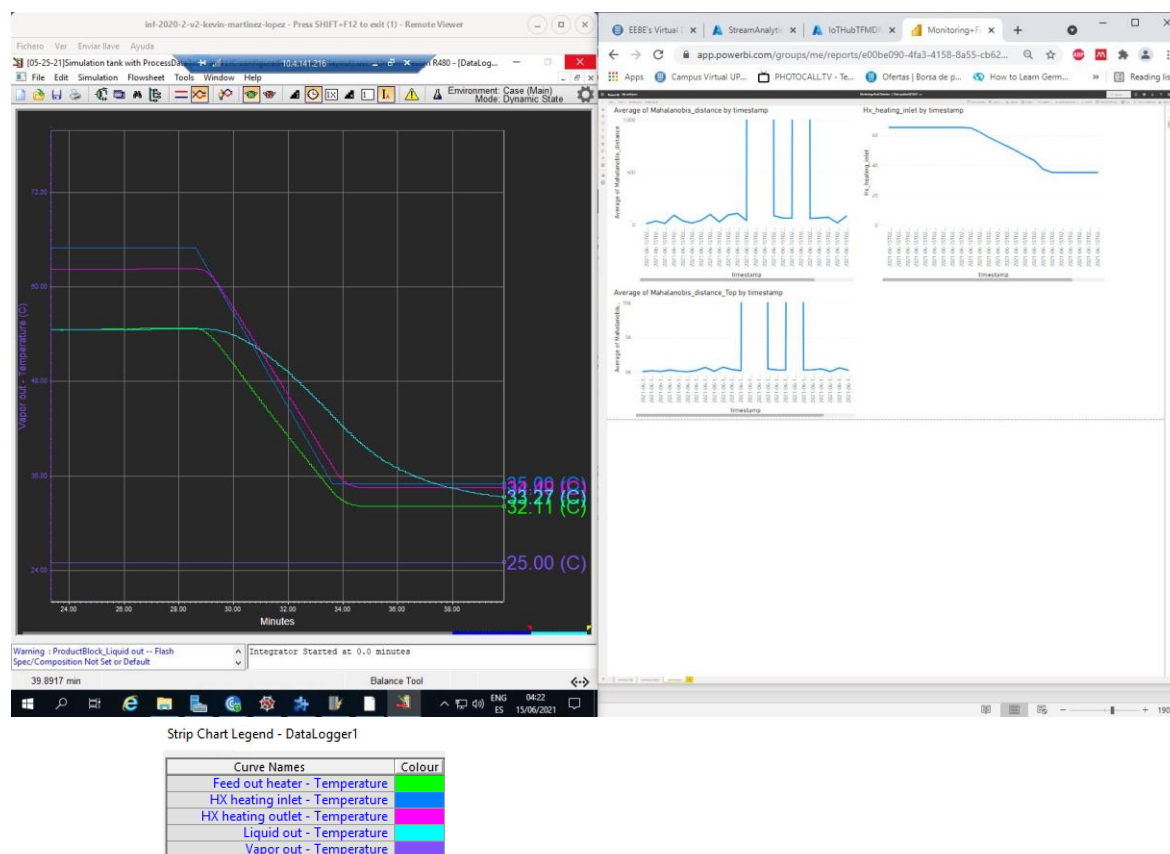


Figure 59. On the left side, it is shown Datalogger 1, where the temperature of HX heating inlet is graphed over time while UniSim is running together with the Python code and the modular communications architecture. The left side corresponds to all the programs that are being run in a remote computer located at the University, to which is accessed with a remote connection. On the right side, it is shown the Fault Detection Palette, allocated in the Cloud, which is being accessed from the personal computer of the student at home.

1.20.2. Analysis of the results obtained during the proof of concept

In this subsection, it is intended to compare the evolution of the variable HX heating inlet-temperature that can be followed with both the use of the Datalogger tool available at UniSim and the use of the Fault Detection Palette available at the Cloud.

On the other hand, it is pretended to make an analysis of the evolution experimented by the temperature of the stream HX heating inlet, and the mean and top values of the Mahalanobis distance in order to validate the use of the modular communications architecture together with the Data Analysis and Machine Learning Algorithms for fault diagnosis.

1.20.2.1. Evolution HX heating inlet-temperature followed with the Datalogger

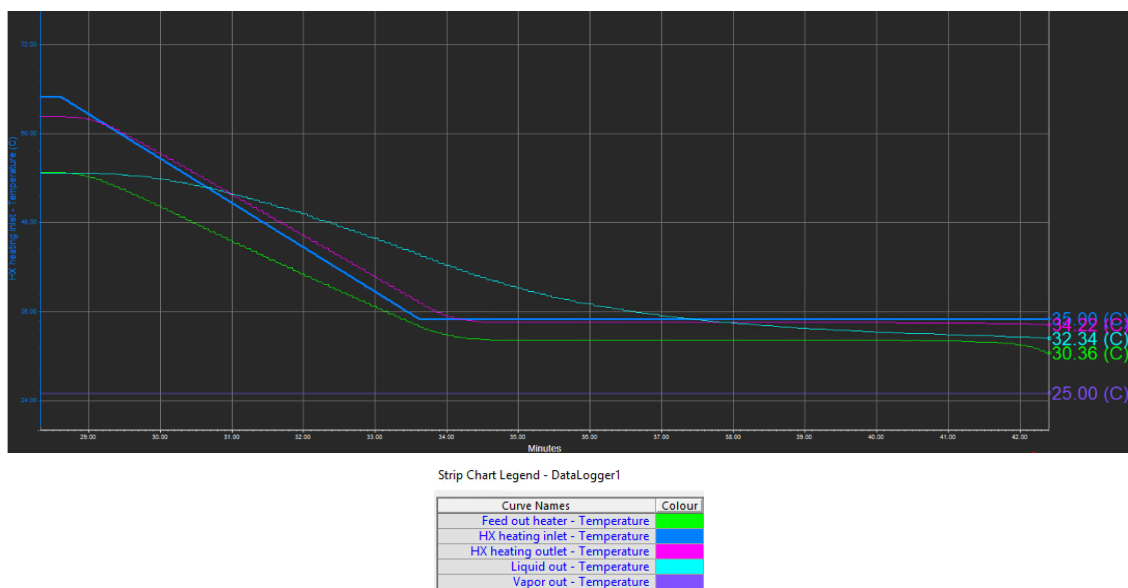


Figure 60. Datalogger 1, where the evolution of the variable HX heating inlet-temperature can be studied. It is possible to see that the variable starts at 65°C and ends up at 35°C in an interval of 5 minutes, which corresponds to the application of a negative temperature ramp of -6°C/min. The thickness corresponding to the curve of the variable has been increased for its better visualization.

1.20.2.2. Evolution of the HX heating inlet-temperature and of the Mahalanobis distance followed from the Fault Detection Palette

The analysis of the evolution experienced by the temperature of the stream HX heating inlet, and the mean and top values of the Mahalanobis distance is to be performed from the use of several snapshots. These snapshots were taken to each of the graphics from the Fault Detection Palette, where the evolution of these variables was recorded, while the simulated case was running.

It is important to keep in mind the threshold value with outliers for the Mahalanobis distance calculated by the machine learning algorithm. This value corresponds to 44 as it is seen in Figure 51.

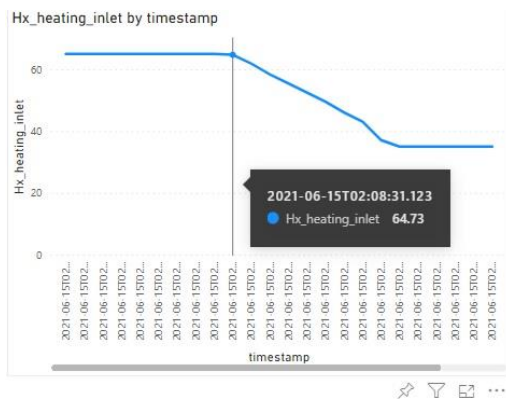


Figure 61. Graphic for the temperature of the stream HX heating inlet. It is appreciated that the temperature ramp started a few seconds ago.

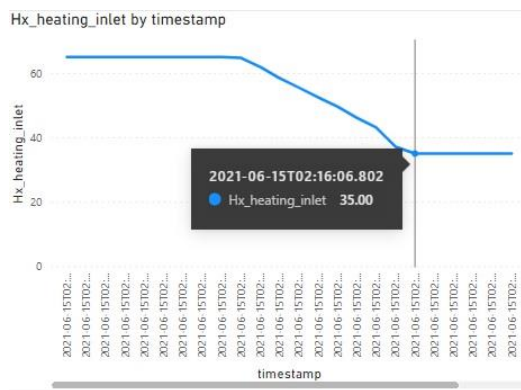


Figure 62. Graphic for the temperature of the stream HX heating inlet. It is appreciated that the temperature ramp has finished.

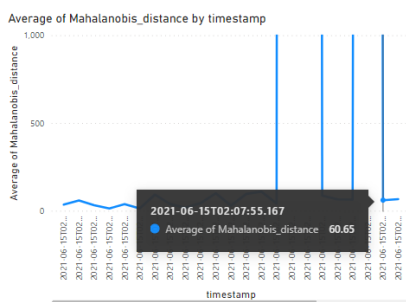


Figure 63. Graphic for the mean value of the Mahalanobis distance. It is possible to appreciate that the mean value shown was recorded by the time the temperature ramp was starting. The value is above the threshold value.

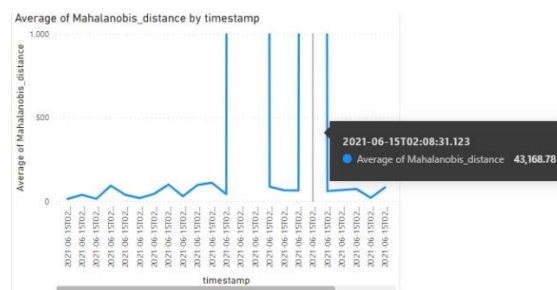


Figure 64. Graphic for the mean value of the Mahalanobis distance. By the time of the record, the temperature ramp had started a few seconds ago. The mean value for the indicator has greatly surpassed the threshold value. The fault is detected before it provokes significant changes on the status of the plant.

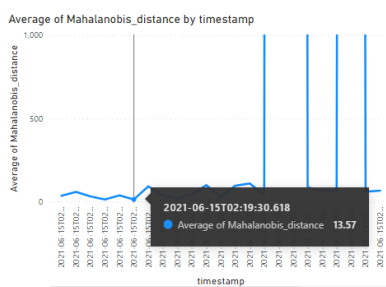


Figure 65. Graphic for the mean value of the Mahalanobis distance. By the time of the record, the temperature ramp had finished three minutes ago. The mean value of the indicator is below the threshold value, but the fault is still present. The temperature is still thirty degrees below its original value.

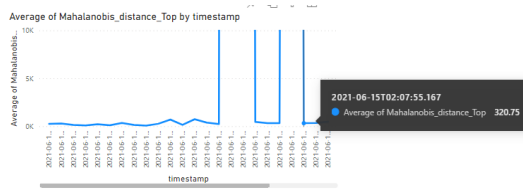


Figure 66. Graphic for the top value of the Mahalanobis distance. The value was recorded by the time the temperature ramp was starting

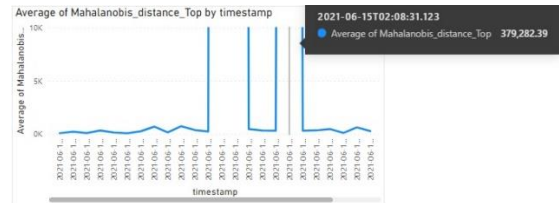


Figure 67. Graphic for the top value of the Mahalanobis distance. By the time of the record, the temperature ramp had started a few seconds ago. It is observed that the top value is more sensitive than the mean value.

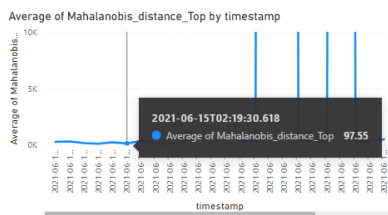


Figure 68. Graphic for the top value of the Mahalanobis distance. By the time of the record, the temperature ramp had finished three minutes ago. The top value is above the threshold value, but it is always necessary to establish decisions according to the mean value. The evaluation of the top value is limited to the situations in which the mean value has also surpassed the threshold.

After the analysis performed, it is possible to state that the monitoring tools for the detection of abnormal events in a simulated plant work and are valid as long as it is remembered the limitations these tools have:

- Any change on a set-point is to be interpreted as a fault as it was concluded in Section 1.18. The changes on set-points usually are ordered in chemical plants as a way to respond to the needs of the process. However, the tools developed in the project are not able to distinguish between an intentional change on a set-point and faults within the plant that provoke unintentional changes on the set-points. Therefore, the use of these tools should be avoided while a set-point, ordered intentionally by the user, is taking place.
- These tools can detect the fault during its starting and while it is happening, when the plant is at transient regime, but they stop working after the plant stabilizes independently of the disappearance or not of the fault. The aforementioned statement is appreciable in Figure 64.

Another point to bear in mind is that these tools have been proved to identify the presence of a fault before the status of the plant is changed significantly. This permits the user to react with more time, anticipate the change and take the corrective measures required for reverting the change on the status of the plant, if possible.

1.21. Guidelines for future development

In this section, some guidelines are to be provided so as to continue working around the ideas concluded at the end of this project. The purpose of these guidelines is to provide new ideas for another person to further develop the current capacities of the prototype explained in this project.

The current monitoring tools for the detection of faults, although being functional, have limitations related to its use. Furthermore, the addition of new functionalities to the tools or the improvement of current functionalities could be explored.

Aiming to improving the monitoring tools, add new functionalities to the monitoring tools or improve existing ones, the following ideas are proposed:

To post the data sent to the Cloud in web services for its display to third-parties.

To use the threshold value calculated by the Machine Learning Algorithm as an alarm in more advanced setups of the monitoring tools in future projects.

The use of Azure Time Series Insight service available at the Cloud. This service can read on-line data from the process and get information from the temporal series of data (31).

To explore the possibility of uploading the Machine Learning and the Data Analysis Algorithms to the Cloud with the use of Microsoft Azure Machine Learning Studio, which is a service that permits to develop the aforementioned algorithms directly or to store them in the Cloud (32).

Related to the performance of the Machine Learning Algorithm developed in this project, some ideas could be explored for further developing and improving its performance and capabilities:

To explore the sensitivity of the indicator towards different intensities on the fault generated. That is to say, to figure out if the indicator shows different results as a function of the magnitude associated with the faults produced: huge or minor faults.

To explore the sensitivity of the indicator towards different faults' nature. That is to say, is it possible to classify the different kinds of faults with this indicator? Are required additional indicators such as the variances related to the principal components, or others? Is it required to train the algorithm differently?

To try to develop and perform a more sophisticated training of the algorithm based on the use of a dataset of temporal series of data aimed at the classification of faults based on its nature.

In this project, the algorithm is trained with a unique temporal series (dataset), while its performance is tested with a new temporal series (online-data).

To try to construct a dataset composed of data points, where the data points are a whole temporal series of data. Each datapoint would be assigned a tag named after the nature of the fault or the absence of faults present during the record of the temporal series of data. Once the dataset is obtained, to train a classifier with this dataset.

Conclusions

A modular communications architecture has been successfully developed and its performance relative to data monitoring and data displaying has been validated.

A functional prototype that employs data analysis and machine learning algorithms together with the aforementioned communications architecture for on-line fault diagnosis in simulated plants has been built-up. However, the prototype developed has its limitations, which implies that the detection of faults is restricted to certain hypotheses. Concretely, it is required for the chemical plant not to suffer intentional changes on the set-points, or if suffered, the results of the model are not to be considered until the simulated plant is at steady-state conditions again.

The validation of both the modular communications architecture and the data analysis and machine learning algorithms proves the potential associated with these new technologies when they are combined and the consequent gain of knowledge, relative to the status of a chemical plant, that can be obtained. It has been validated not only the capacity to detect the presence of a fault, but it has been proved that the prototype detects the presence of the fault before it provokes significant changes on the status of the plant. In other words, the use of these tools allows the prediction of significant changes on the status of the plant, permitting the user to anticipate these significant changes and try to minimize harmful consequences to the plant.

After the validation documented in this project, it can be concluded that companies will be gradually more receptive to a change in the paradigm on how to manage the data they generate as similar or better results from future investigations are published. Instead of storing the data locally, companies will gradually be more open-minded to use the new technologies for getting insights from the data they generate, for ultimately improving their operations and activities. This prediction is not only limited to the chemical industry, but to any industrial sector, which generates or operates with Big Data.



Economic Analysis

This section is devoted to perform the economic analysis of the project, which is to be broken down as a function of the different kinds of costs incurred.

1.22. Electricity costs

It is considered the electricity consumed by the desktop computers used during the project. On the one hand, the use of personal computer has been required for all the tasks. On the other hand, it has been required the use of a remote computer in the stage related to the implementation of the prototype.

On the one hand, the price for the electricity costs is based on the average price of 0.0943 per kWh from 2020 in Spain, considering that the user has a regulated tariff, which is reported at (33). On the other hand, the average power consumption by the computers has been considered as the summation of the power consumed by the computer (190 W) and the one consumed by the monitor (220 W) according to (34).

The cost associated with each activity developed with the personal computer is indicated in Table 1.

Table 1. Electricity costs incurred by the use of the personal computer

Concept of payment	Quantity (hours)	Consumption (kWh)	Price (€/kWh)	Cost (€)
Bibliographic research	40	16,4	0,0943	1,5
Implementation of the prototype	270	110,7	0,0943	10,4
Report writing	90	36,9	0,0943	3,5
Report review	35	14,35	0,0943	1,4
Weekly meetings	15	6,15	0,0943	0,6
Total cost				17,4

The remote computer is considered to have the monitor disconnected since it has been used from home. Therefore, the power consumption only accounts for the computer (190 W).

The cost relative to the use of the remote computer is shown in Table 2.

Table 2. Electricity costs incurred by the use of the remote computer

Concept of payment	Quantity (hours)	Consumption (kWh)	Price (€/kWh)	Cost (€)
Implementation of the prototype	270	51,3	0,0943	4,8
Total cost (€)				4,8

1.23. Cloud Platform costs

The Cloud Platform employed (Microsoft Azure) offers different metered services, the use of which follows the pay-per-use policy. In other words, the user is billed as per the intensity on the utilization of the services. Three services from the Microsoft Azure Cloud have been used:

1. Azure IoT Hub. This service has been used in its free tier version, which is limited to the number of devices connected (500 devices) and the number of messages that can be received (8000 messages/day).
2. Azure Stream Analytics. This service is metered according to:
 - a. its scalability (number of devices using the service and number of streaming units)
 - b. the intensity of use (number of hours of use associated with each streaming unit)
3. Power BI. This service has been used in its free version available to students.

The information relative to the prices of each service offered by the Azure Cloud Platform can be found at (35).

Since Azure Stream Analytics is the only service that costs money, the costs incurred due to its utilization are shown as a function of the intensity of use in Table 3 and as a function of the scalability imposed to the service in Table 4.

Table 3. Costs relative to the intensity of use of the Azure Stream Analytics service

Concept of payment		Streaming units	Quantity (hours)	Price (€/h-streaming units)	Cost (€)
Azure Stream Analytics	Intensity of use	3	12	0,090	3,25
Total cost (€)					3,25

Table 4. Costs relative to the scalability of the Azure Stream Analytics service

Concept of payment		Devices	Quantity (months)	Price (€/device-month)	Cost (€)
Azure Stream Analytics	Scalability	1	3	1	2,46
Total cost (€)					2,46

1.24. Personnel costs

The personnel costs are related to the need for hiring a person able to perform the project. This person has been considered a graduate of the second university cycle according to the Collective agreement for the engineering company sector and technical studies offices for the year 2020 available at (36).

Additionally, the report has been reviewed by two graduates of the third university cycle according to the same Collective agreement. Furthermore, weekly meets of forty minutes on average have been performed between the three individuals so as to review the state of the project and plan how to proceed on the realization of the project.

For both a graduate of the second university cycle and a graduated third university cycle, the annual salary is the same, being 23973.88 €/year, according to. Considering that a year comprises fifty-two weeks, the price per hour to pay to a graduate is 11.52 €/year. Bearing this in mind, the costs relative to hiring personnel are displayed in Table 5.

Table 5. Costs relative to the personnel hiring

Concept of payment	Quantity (hours)	Personnel	Price (€/h)	Cost (€)
Bibliographic research	40	1	11,52	460,8
Implementation of the prototype	270	1	11,52	3110,4
Report writing	90	1	11,52	1036,8
Report review	35	3	11,52	1209,6
Weekly meetings	15	3	11,52	518,4
Total cost (€)				6336

1.25. Total costs

All the costs classified by its nature are summarized in Table 6, which also shows the total cost related to the realization of the project.

Table 6. Total cost of the project

Concept of payment	Cost (€)
Personal computer (Electricity costs)	17,4
Remote computer (Electricity costs)	4,8
Cloud Platform costs (intensity of use)	3,25
Cloud Platform costs (scalability)	2,46
Personnel costs	6336
TOTAL (€)	6363,9



Webography

1. Digital twin - Wikipedia. [online]. [Accessed 16 June 2021]. Available from: https://en.wikipedia.org/wiki/Digital_twin
2. List of chemical process simulators - Wikipedia. [online]. [Accessed 16 June 2021]. Available from: https://en.wikipedia.org/wiki/List_of_chemical_process_simulators
3. UniSim – Software for Process Design and Simulation. [online]. [Accessed 16 June 2021]. Available from: <https://www.honeywellprocess.com/en-US/explore/products/advanced-applications/unisim/Pages/default.aspx>
4. UniSim®Design Suite: Simple, Powerful, Flexible. [online]. [Accessed 16 June 2021]. Available from: https://www.honeywellprocess.com/en-US/online_campaigns/unisim-design/Pages/products.html#proj0
5. Why Use Python? - Full Stack Python. [online]. [Accessed 16 June 2021]. Available from: <https://www.fullstackPython.com/why-use-Python.html>
6. DOCUMENTATION TEAM, AspenTech. *HYSYS ® 2004.2 Customization Guide* [online]. 2004. [Accessed 16 June 2021]. Available from: <http://www.aspentech.com>
7. COM, DCOM, and Type Libraries - Win32 apps | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/en-us/windows/win32/midl/com-dcom-and-type-libraries>
8. Understanding Objects, Properties, and Methods | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/en-us/office/vba/word/concepts/objects-properties-methods/understanding-objects-properties-and-methods#what-is-a-property>
9. Graphical user interface - Wikipedia. [online]. [Accessed 16 June 2021]. Available from: https://en.wikipedia.org/wiki/Graphical_user_interface
10. Python - GUI Programming (Tkinter) - Tutorialspoint. [online]. [Accessed 16 June 2021]. Available from: https://www.tutorialspoint.com/Python/Python_gui_programming.htm
11. Paessler - The Monitoring Experts. [online]. [Accessed 16 June 2021]. Available from: <https://www.paessler.com/>
12. OPC UA Server Simulator - Simulate real-time and historical data. [online]. [Accessed 16 June 2021]. Available from: <https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-server-simulator/>
13. DataFEED OPC Suite - Softing | Softing. [online]. [Accessed 16 June 2021]. Available from: <https://data-intelligence.softing.com/es/productos/opc-software-platform/DataFEED-opc-suite/#tx-dftabs-tabContent2>

14. Introducción a Azure IoT Hub | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/es-es/azure/iot-hub/about-iot-hub>
15. Install and use Azure IoT explorer | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/en-us/azure/iot-pnp/howto-use-iot-explorer>
16. Introducción a Azure Stream Analytics | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/es-es/azure/stream-analytics/stream-analytics-introduction>
17. IoT Hub | Microsoft Azure. [online]. [Accessed 16 June 2021]. Available from: <https://azure.microsoft.com/en-us/services/iot-hub/>
18. Use the Azure portal to create an IoT Hub | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal>
19. Real-time data visualization of data from Azure IoT Hub – Power BI | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-power-bi>
20. Azure Stream Analytics | Microsoft Azure. [online]. [Accessed 16 June 2021]. Available from: <https://azure.microsoft.com/en-us/services/stream-analytics/>
21. Streaming Data Pipelines: Building a Real-Time Data Pipeline Architecture. [online]. [Accessed 16 June 2021]. Available from: <https://www.precisely.com/blog/big-data/streaming-data-pipelines-how-to-build-one>
22. Setting up an End-to-End Data Streaming Pipeline | 6.3.x | Cloudera Documentation. [online]. [Accessed 16 June 2021]. Available from: https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/kafka_end_to_end.html
23. Azure Stream Analytics Pivoting Rows To Columns - Stack Overflow. [online]. [Accessed 16 June 2021]. Available from: <https://stackoverflow.com/questions/49425209/azure-stream-analytics-pivoting-rows-to-columns>
24. Data Visualization | Microsoft Power BI. [online]. [Accessed 16 June 2021]. Available from: <https://powerbi.microsoft.com/en-us/>
25. ¿Qué es Power BI? | Deloitte España. [online]. [Accessed 16 June 2021]. Available from: <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-power-bi.html>
26. Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE. [online]. [Accessed 16 June 2021]. Available from: https://www.cienciadedatos.net/documentos/35_principal_component_analysis

27. Mahalanobis Distance: Simple Definition, Examples - Statistics How To. [online]. [Accessed 16 June 2021]. Available from: <https://www.statisticshowto.com/mahalanobis-distance/>
28. What is West Bengal's biggest contribution to the world? - Quora. [online]. [Accessed 16 June 2021]. Available from: <https://www.quora.com/What-is-West-Bengals-biggest-contribution-to-the-world>
29. How to use machine learning for anomaly detection and condition monitoring | by Vegard Flovik | Towards Data Science. [online]. [Accessed 16 June 2021]. Available from: <https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7>
30. Machine learning for anomaly detection and condition monitoring | by Vegard Flovik | Towards Data Science. [online]. [Accessed 16 June 2021]. Available from: <https://towardsdatascience.com/machine-learning-for-anomaly-detection-and-condition-monitoring-d4614e7de770>
31. Time Series Insights | Microsoft Azure. [online]. [Accessed 16 June 2021]. Available from: <https://azure.microsoft.com/es-es/services/time-series-insights/>
32. ¿Qué es Azure Machine Learning Studio? | Microsoft Docs. [online]. [Accessed 16 June 2021]. Available from: <https://docs.microsoft.com/es-es/azure/machine-learning/overview-what-is-machine-learning-studio>
33. ¿Cuánto cuesta la luz al mes? Consumo y gasto medio €/mes. [online]. [Accessed 16 June 2021]. Available from: <https://tarifasgasluz.com/faq/cuanto-cuesta-luz-mes>
34. ¿Cuánto Consumo el Ordenador? Sal de dudas. [online]. [Accessed 16 June 2021]. Available from: <https://ganaenergia.com/blog/que-consumo-nos-supone-utilizar-el-ordenador/>
35. Calculadora de precios | Microsoft Azure. [online]. [Accessed 16 June 2021]. Available from: https://azure.microsoft.com/es-es/pricing/calculator/?&ef_id=CjwKCAjwn6GGBhADEiwAruUcKu5_6MduMR38BeGflnk6ZAYG93SJhJkXUEGNzrNalxZ2rU6CboCu-xoCk7wQAvD_BwE:G:s&OCID=AID2100112_SEM_CjwKCAjwn6GGBhADEiwAruUcKu5_6MduMR38BeGflnk6ZAYG93SJhJkXUEGNzrNalxZ2rU6CboCu-xoCk7wQAvD_BwE:G:s&gclid=CjwKCAjwn6GGBhADEiwAruUcKu5_6MduMR38BeGflnk6ZAYG93SJhJkXUEGNzrNalxZ2rU6CboCu-xoCk7wQAvD_BwE
36. BOE.es - BOE-A-2019-14977 Resolución de 7 de octubre de 2019, de la Dirección General de Trabajo, por la que se registra y publica el XIX Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos. [online]. [Accessed 16 June 2021]. Available from: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2019-14977