

University of Wales Trinity Saint David
School of Engineering



Mechanical and Manufacturing Engineering

Design and Implementation of a Real Time
Energy Monitoring System and Dashboard for
a Festo CP Lab Station

Sheunesu John Ziki

2205423

2026

Declaration of authenticity

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: Sheunesu John Ziki

Date: 24/04/2026

Acknowledgement of the use of AI tools

I acknowledge the use of Perplexity (<https://www.perplexity.ai/>) on 01/10/2025 to research.

I acknowledge the use of ChatGPT (<https://chatgpt.com/>) on 23/04/2026 to research.

I confirm that I have not presented any AI generated text as my own work. I confirm I have copies of my drafts, notes and other resources I used before, and after the use of AI tools which I may be asked to provide in evidence.

Abstract

This project developed and evaluated a real time energy monitoring system for a Festo CP Lab station. The system integrated electrical, pneumatic and environmental sensors with OPC UA communication and a Node-RED dashboard to provide live monitoring of energy and process behaviour. Experimental tests showed that the station has a significant standby energy demand. Overloaded conveyor, blocked exhaust and robot speed changes produced measurable differences in the systems responses. The results also showed that pneumatic pressure stayed stable across pallet count variation while airflow was more sensitive to abnormal operating conditions. Faster robot joint speed reduced the energy consumption per phone by shortening the cycle time. The project demonstrates that a low cost, multi sensor monitoring system can support energy analysis, process observation and fault detection in automated manufacturing environments.

Acknowledgments

I want to give my deepest thanks to my supervisor Jordan Jenkins for his incredible guidance and support throughout this project. His patience, practical advice, and ability to push me in the right direction made all the difference.

To my family thank you for your patience, encouragement, and constant support throughout my studies. Your understanding, motivation and prayers do not go unnoticed.

Table of Contents

1	Introduction	7
1.1	Background	7
1.2	Problem Statement.....	7
1.3	Aim	7
1.4	Objectives.....	7
1.5	Research Questions	8
1.6	Scope	9
2	Literature Review	10
2.1	Industry 4.0 and Smart Manufacturing	10
2.2	Energy Monitoring Systems in Manufacturing	11
2.3	Integration of OPC UA Communication.....	12
2.4	Real Time Monitoring and Dashboard Design.....	12
2.4.1	Dashboard Software	13
2.5	Festo CP Lab as a Research Platform.....	13
2.6	Evaluation and Usability of Monitoring Systems.....	14
2.6.1	Sensor Systems for Air and Electricity Monitoring	14
2.7	Six sigma and data quality in monitoring systems.....	15
2.8	Scalability and future trends.....	16
2.9	Literature gaps.....	16
3	Methodology.....	17
3.1	System architecture	17
3.2	Hardware selection and implementation	18
3.2.1	Experimental platform	18
3.2.2	Sensor selection and justification	19
3.3	Data acquisition system	22
3.3.1	Arduino data processing	22
3.3.2	Sampling method	22
3.3.3	Serial communication.....	22

3.3.4	Post processing	23
3.4	OPC UA integration	23
3.5	Node-RED dashboard	24
3.5.1	System integration architecture	24
3.5.2	Node-RED flow execution	24
3.5.3	Dashboard design and visualisation	25
3.5.4	Design justification	27
3.5.5	Experimental testing role	27
3.5.6	Limitations	27
3.6	Data logging and storage	28
3.7	Experimental procedure	28
3.8	Data analysis methods	29
3.9	Assumptions	30
3.10	Limitations	30
3.11	Summary	30
4	Results	31
4.1	Results overview	31
4.2	Experimental test matrix	31
4.3	Baseline and idle consumption	33
4.4	Conveyor loading and overload behaviour	34
4.5	Pneumatic supply pressure behaviour	35
4.6	Exhaust airflow behaviour	37
4.7	Full production cycle energy usage	39
4.8	Effect of removing the fuses	41
4.9	Data quality and filtering	43
4.10	Summary	44
5	Discussion	45
5.1	Overview of key findings	45
5.2	Electrical energy behaviour	45

5.3	Pneumatic system behaviour	46
5.4	Effects of process changes	46
5.5	EMS performance evaluation	47
5.6	Data quality and signal processing	48
5.7	Industrial implications	48
5.8	Limitations of the study	49
5.9	Summary	49
6	Conclusions and future work	50
6.1	Conclusions	50
6.2	Main contributions	50
6.3	Recommended further work	51
6.4	Final remarks	51
7	References	52
8	Appendix	54
8.1	Node-RED flow diagrams	54
8.2	Arduino code	54
8.3	Python code	58
8.4	Data filtering code	66
	Figure 1: Visualisation of industry 4.0 smart factories	10
	Figure 2: Shows industry 4.0 structure.	11
	Figure 3: Visualisation of Energy Monitoring Systems (avatech.com).	12
	Figure 4: Shows an example of a dashboard.	13
	Figure 5: Shows the Festo CP Lab	14
	Figure 6: Shows DMAIC methodology	15
	Figure 7: Shows the systems data flow.	17
	Figure 8: Shows Arduino and sensor configuration.	18
	Figure 9: Shows the Festo automated production cell.	19
	Figure 10: Shows the INA219 current sensor module.	19
	Figure 11: Shows the 100PSI transducer.	20
	Figure 12: Shows the FS3000-1005 air velocity sensor	20

Figure 13: Shows the SDP610-500Pa differential sensor.	21
Figure 14: Shows the BME 280 environmental monitoring sensor.	21
Figure 15: Shows the Arduino Uno microcontroller.	22
Figure 16: Shows the Node-RED flow diagram with OPC UA client, inject nodes and dashboard connections.	25
Figure 17: Shows the dashboard layout with real time gauges and time series plots.	26
Figure 18: Shows the dashboard layout with real time gauges and time series plots.	27
Figure 19: Show the Full production run at 50% joint speed showing AC power variation over time.	31
Figure 20: Shows a comparison of idle and conveyor only AC profiles.	34
Figure 21: Conveyor current under normal operation and under additional loading.	35
Figure 22: Shows pneumatic supply pressure during 1 pallet and 10 pallet tests.	37
Figure 23: Shows airflow behaviour during blocked and normal tests.	38
Figure 24: Energy consumption per phone as a function of robot joint speed.	40
Figure 25: Production efficiency expressed as phones per kWh against robot joint speed. .	41
Figure 26: Shows a comparison of energy consumption per phone for normal run and nor fuses run.	42
Figure 27: Shows a comparison of raw and filtered conveyor current.	44
Table 1: Showing sensor types and uses.	15
Table 2: Sensor specifications.	21
Table 3: Shows developed experimental procedure.	28
Table 4: Shows experimental test matrix.	32
Table 5: Shows the idle and baseline results.	33
Table 6: Shows conveyor load comparison.	35
Table 7: Shows the pneumatic pressure comparison.	36
Table 8: Shows the exhaust airflow comparison.	38
Table 9: Shows the full run energy comparison of the different tests.	40
Table 10: Shows robot speed comparison tests.	40
Table 11: Shows raw results versus filtered current.	43
Equation 1: Power formula (Nilsson, J. W. and Riedel, S. A. (2019)).	23
Equation 2: Electrical energy equation.	29

1 Introduction

1.1 Background

Industry 4.0 has transformed manufacturing by integrating automation, connectivity and data driven decision making into production plants (May *et al.*, 2017a). The pursuit of better efficiency and lower environmental impact has made real time monitoring more important for understanding process behaviour and energy consumption. Energy monitoring systems (EMS) and digital dashboards support this objective by providing visual insights into electrical and pneumatic behaviour, by enabling for more informed operational decisions . The Festo CP Lab provides a modular educational platform for studying these concepts in a controlled automation environment but does not provide an integrated view of energy and process data. This gives room to develop a monitoring solution that links machine behaviour with resource usage.

1.2 Problem Statement

The absence of a combined energy monitoring system and visualisation dashboard of energy usage limits the ability to analyse efficiency and sustainability within the Festo CP Lab station. Although the CP Lab provides access to process variables such as the valve positions and actuator states, they are not linked to real time energy usage. This means that the operators cannot easily evaluate how the process conditions influence the air and electricity consumption. This is a compromise to the lab's potential as a learning and research platform.

1.3 Aim

The aim of the project is to design and implement a real time dashboard and energy monitoring system for the visualisation of air and electricity usage for the Festo CP Lab station.

1.4 Objectives

1. Identify a range of sensors to measure pneumatic or electricity usage in the CP lab station.
 - Portable with all essential components.
 - Universal power and data system.
 - Self-calibrating or easy calibration.
 - Dashboard that can be configured to work remotely on a different device type.
2. Acquire and process existing process data from the open platform communications (OPC) enabled Festo system.

- Include both real time data acquisition and historical data to allow for a deeper trend analysis.
 - Identify the process data to be acquired for example actuator status, valve positions, and cycle times.
 - Use Six Sigma techniques such as data filtering, normalisation, and error handling for data quality before we use it.
 - Specify the required OPC standard detailing the planned validation steps to ensure accuracy and reliability of data stream.
3. Develop a real time dashboard using Node-Red or MATLAB/Python to display both process and energy data in a user-friendly way.
 - Requirements for the dashboard layout such as responsive design and customisable views.
 - Dashboard functionalities like trend charts, real time status indicators and alert systems for any abnormal conditions.
 - Validation steps such as usability testing.
 - Ensure the dashboard can easily incorporate new sensors or data types for future proofing.
 4. Demonstrate the system with case studies, which will show how energy usage relates to process efficiency.
 - Criteria for case studies such as process modes or loading conditions.
 - Quantitative metrics for process efficiency such as cycle time or any quantity measures alongside energy consumption.
 - A controlled experiment to isolate variables affecting energy use and efficiency.
 - Documentation of energy saving opportunities.
 5. Evaluate the systems effectiveness in improving the monitoring, sustainability assessment, and operator decision making.
 - Monitoring – accuracy, data latency, and system uptime.
 - Sustainability – ability to find energy saving opportunities, track consumption trends, and report the environmental impact.
 - Operator decision making – measuring improvements in decision speed, error reduction, or process adjustments based on dashboard insights.
 - Plans for collecting quantitative data such as user feedback.
 - Scalability assessment. How it can be expanded to other lab stations.
 - Iterative refinement cycles from evaluation outcomes.

1.5 Research Questions

1. How can air and electrical energy consumption be monitored in real time?
2. How can a dashboard's design improve the usability and efficient decision making for operators?
3. What measurable improvements can be made in the sustainability assessment through integrated real time monitoring?

1.6 Scope

This project will focus on a single Festo CP Lab station. Implementation of two to three sensors for the air flow and electricity consumption. The system will be designed to be expandable, but advanced analytics and predictive maintenance are beyond this project's immediate scope.

2 Literature Review

The fast evolution of manufacturing technology has transformed traditional production into smart data driven ecosystems. With a push towards sustainable manufacturing Industry 4.0 has given a rise to smart manufacturing environments through automation, data exchange, and cloud-based connectivity. Due to this, integrating real time process and energy monitoring has become essential to improving efficiency and in reducing environmental impact. The recent advances in Open Platform Communications (OPC UA), Internet of Things (IoT), and cyber physical systems have made these capabilities possible in both industrial and academic settings.

This chapter will review existing research relevant to the project. It will begin with the background research of industry 4.0 and sustainable manufacturing, followed by research into energy monitoring systems, Festo CP Lab systems and OPC communication, dashboard technologies, and sensor applications. The literature review informed the methodology by guiding the sensor selection, communication protocols, methods and visualisation tools used in the systems design.

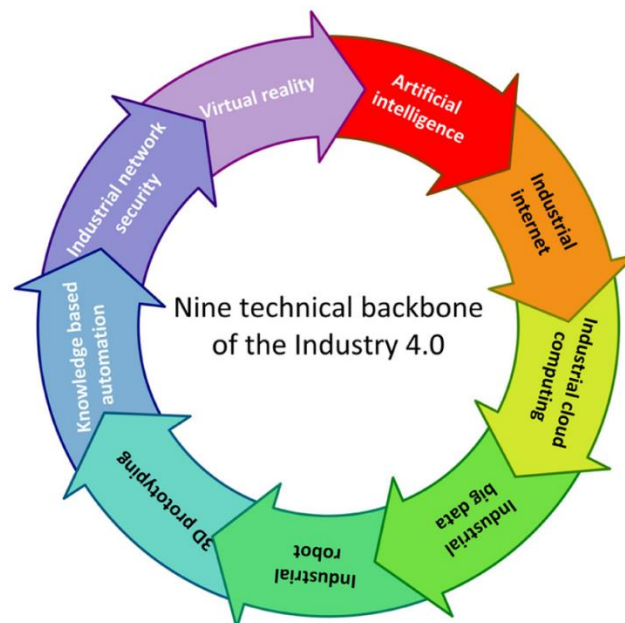


Figure 1: Visualisation of industry 4.0 smart factories.

2.1 Industry 4.0 and Smart Manufacturing

Industry 4.0 represents the digital transformation of manufacturing systems through cyber physical systems (CPS), industrial IoT (IIoT), big data analytics and artificial intelligence. Smart factories integrate sensors, actuators and controllers into networked systems that communicate via open protocols such as OPC UA, Modbus or MQTT. These networked systems allow for real time decision making, predictive maintenance and self-optimisation.

Most Industry 4.0 architecture follows a structure which consists of the following as seen in Figure 2 below:

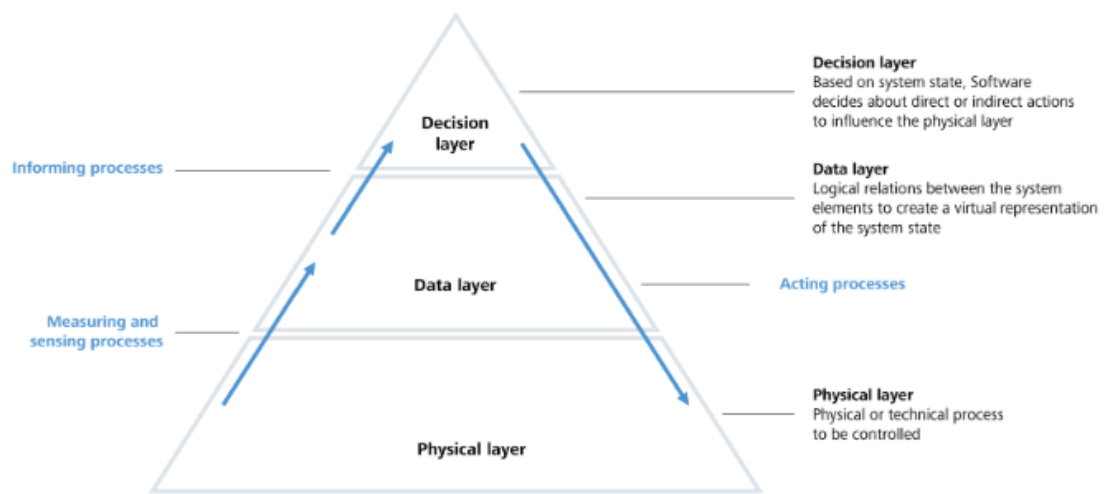


Figure 2: Shows industry 4.0 structure.

From Figure 2 above shows the physical layer which represents the sensors and actuators that feed information to the microcontrollers (data layer), which transmits the information to the dashboard (decision layer).

Research shows that dashboards play a very pivotal role in bridging the gap caused by data interpretation. Studies from Bosch Rexroth (2022) and Siemens (2023) show that effective data presentation on dashboards reduces industry downtime by approximately 20% and increase the operator's responsiveness by 30%.

2.2 Energy Monitoring Systems in Manufacturing

Depending on the sector, it has been seen that energy is one of the biggest costs in manufacturing, accounting for about 15% - 40% (Lee and Lee, 2025). This can be linked to a trend in increasing energy costs. Due to this we saw the introduction of the UK's Industrial Decarbonisation Strategy that emphasises the point of continuously monitoring energy usage to being essential to achieving its Net Zero goals by 2050 (BEIS2021).

Energy monitoring is a very key foundation for operational optimisation and sustainability assessments. A study conducted by Nicla Frigerio has shown that by implementing an energy monitoring system (EMS), it has been easier to track efficiency and hotspots in manufacturing plants (May *et al.*, 2017b). A similar study was also conducted which reviewed a sensor-based energy metering system. This review showed the importance of selecting the most appropriate sensors and data processing frameworks to ensure data accuracy and system scalability (Frigerio, Matta and Rasella, 2022)

From real world production environments, we have seen that the implementation of IoT based sensors for energy efficiency in a beverage factory, achieved real time visibility and enabled proactive decision making that reduced wastage and downtime (Jagtap, Rahimifard and Duong, 2019). These studies show the importance of energy dashboards in industry.

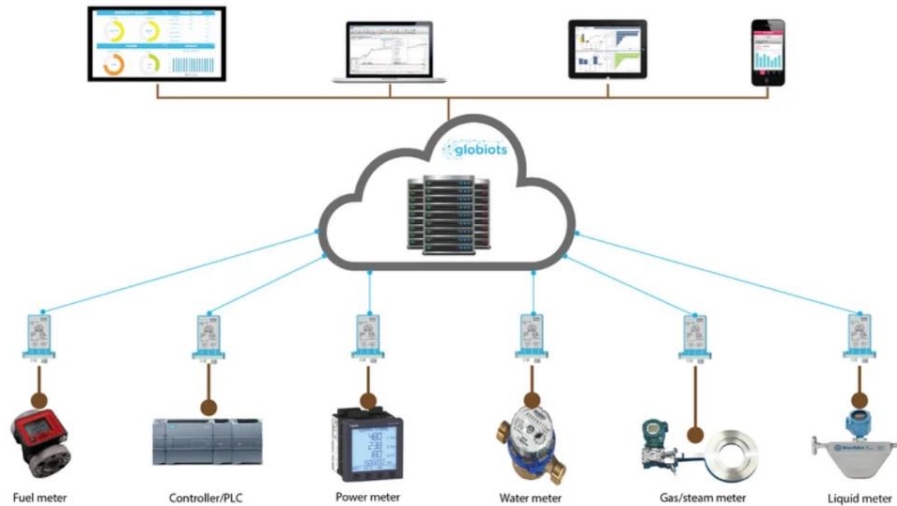


Figure 3: Visualisation of Energy Monitoring Systems (avatech.com).

2.3 Integration of OPC UA Communication

Open Platform Communications Unified Architecture (OPC UA) is widely used in manufacturing because of its platform independent framework that allows for secure and reliable data exchange. Through OPC UA process parameters such as cycle times, valve states, and actuator position can be accessed for monitoring or control. Its flexibility allows for standardised data exchange and secure device connectivity, which gives a platform for digital twins and remote monitoring (Krutwig, 2019). Studies have validated that OPC UA's ability to manage real time energy data streams within manufacturing equipment, has a strong relationship with energy profiled machinery operations (He *et al.*, 2020)

Energy monitoring modules are rarely included in OPC UA, but recent research has advanced OPC UA through automated generation of its information models. This has streamlined the integration process for a wide range of equipment within manufacturing systems. Research has shown why OPC UA is the standard for cross platform communication in EMSs (Busboom, 2024).

2.4 Real Time Monitoring and Dashboard Design

Dashboards are a visual layer used to view data in modern manufacturing industries as shown in Figure 4 below. They convert the numerical data from the sensors into charts, gauges, and alerts to help easily interpret data. For a dashboard to be effective it must be

able to have a balance between information and usability. This allow the operator to interpret the data easily for fast decision making.

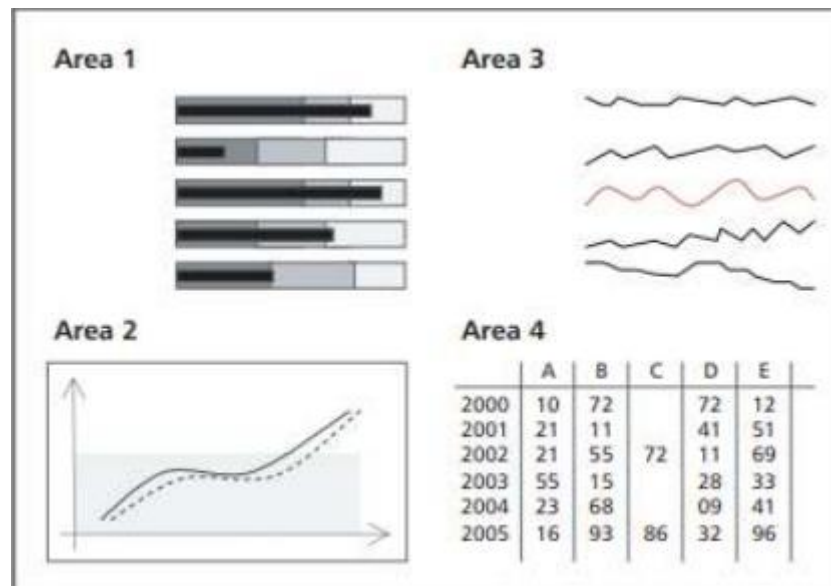


Figure 4: Shows an example of a dashboard.

According to (Hernández, Lehoux and Gosselin), integrating performance data and energy metrics into a dashboard supports intuitive decision making and gives a more transparent view on patterns in energy consumption. Another research paper from (Antus *et al.*) advocated for dashboards to have customisable interfaces, stakeholder specific indicators, and integrated data within real time monitoring systems.

Earlier research such as the Dashboard Demystified (Hetherington), showed that dashboards changed the operator's ability to see any inefficiencies by linking energy consumption data with production output. This enabled better scheduling decisions (Dussault and Electric). These research papers show that real time dashboard's usability and data clarity are especially important to its effectiveness to the operator.

2.4.1 Dashboard Software

1. Node-Red: Is a software developed by IBM which is open source, requires a small amount of code, and is well suited for IoT. It has built in nodes for OPC UA, MQTT, and has easy web deployment.
2. MATLAB App Designer: Integrates seamlessly with data analysis toolboxes but is limited by its deployment abilities compared to web-based dashboard software's.

2.5 Festo CP Lab as a Research Platform

The Festo CP Lab and CP Factory are modular automation platforms that simulate manufacturing industrial processes and environments as shown in Figure 5 below. Their main functions are for research into automation, sustainability, and data backed production

optimisation. According to Festo's website, these systems are meant to replicate real production modules, enable studies of the cyber physical systems, industrial communication, and energy management. Several universities use these systems to teach OPC data acquisition and industry 4.0 integration.

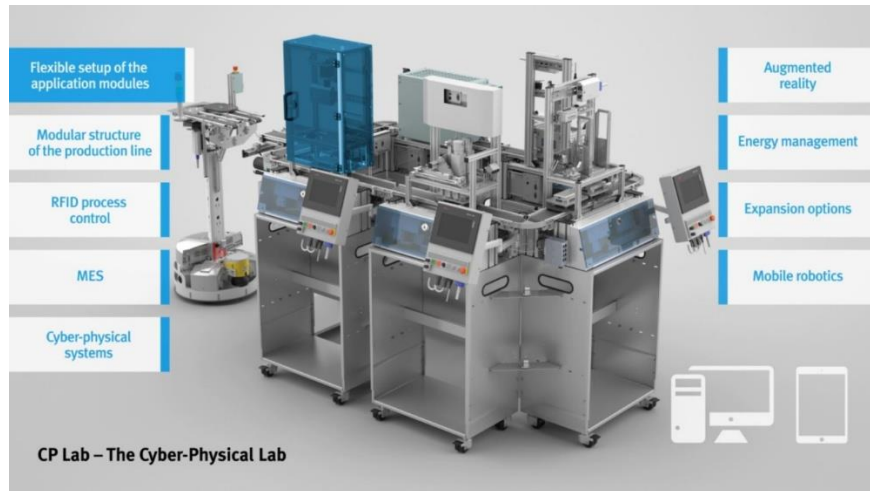


Figure 5: Shows the Festo CP Lab.

The labs provide a practical test bed for connecting hardware sensors, energy meters, and dashboard monitoring. This ensures that experiments conducted are translated to the same scale as industry. The controlled Festo environment provides validation of sustainability tools within a safe system.

2.6 Evaluation and Usability of Monitoring Systems

When it comes to evaluating monitoring systems both qualitative and quantitative data is needed. Research shows the need to assess the data's accuracy, latency, and system reliability alongside user factors such as its ease of use and decision making. Research from (Uhlmann, Polte and Geisert, 2025) proposed an approach to choosing measurement components for production energy monitoring. This was shown by industry implementations like Toyota Motor Uk's energy dashboard, which improved operational understanding and reduced power costs through the use of real time data feedback (Pressac, 2023).

2.6.1 Sensor Systems for Air and Electricity Monitoring

Data accuracy starts with reliable sensing. Some common transducers to monitor air and electricity include:

Table 1: Showing sensor types and uses.

Parameter	sensor	Measurement Range	Example
Air Flow	Renesas F3000 - 1015	0-500L/min	Compressed air monitoring
Electrical Current	Hall effect sensor (ACS712/ACS758)	0-30 A DC/AC	Motor or power draw
Voltage	Voltage divider + ADC	0-250V	Energy measurement

Proper calibration is essential to maintain measurement accuracy. Literature shows the importance of a system that can self-calibrate and filter data to counter any noise (Pertijs, 2014). The integration can be done using microcontrollers such as an Arduino or ESP 32 as done in most research prototypes, due to their low cost and wireless capabilities (Maag, 2018).

2.7 Six sigma and data quality in monitoring systems

Six Sigma allows for a structured methodology such as define, measure, analyse, improve, control (DMAIC) as seen in Figure 6 which improves the process quality (Mittal *et al.*, 2023).

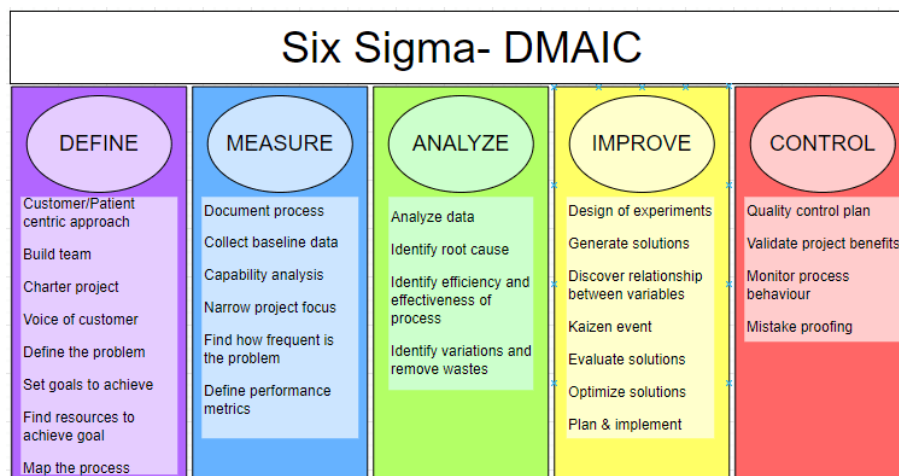


Figure 6: Shows DMAIC methodology.

Six Sigma principles for data acquisition ensure that collected data is accurate, consistent and representative of real-world process behaviour. Methods such as moving average filtering, normalisation and error detection are more important than the visualisation of the data. High quality data is the cornerstone of an effective dashboard, without it, visualisation of the data gives inaccurate insights. This shows that integrating Six Sigma concepts into the data conditioning stage allow for a solid engineering dashboard (Mittal *et al.*, 2023).

2.8 Scalability and future trends

Emerging trends in smart manufacturing point in the direction of digital twins, edge computing and machine learning integration. Dashboards are also evolving from passive displays into diagnostic and predictive tools (Adebowale, 2026). A scalable design, use of modular data structures and open API's ensure longevity and compatibility with future technology.

For educational labs such as the Festo station, scalability allows replication across many stations and remote accessibility for hybrid learning environments. The project's proposed use of Node-Red and python supports for future cloud deployment or connectivity to IoT platforms such as Things Board or Grafana.

2.9 Literature gaps

The literature review shows some key gaps that justify this research such as:

1. Limited integration of real time energy monitoring with dashboard visualisation for Festo CP lab or comparable systems.
2. Few examples of open-source dashboards tailored for manufacturing setups.
3. Insufficient application of Six Sigma quality methods in real time engineering visualisation
4. Need for a modular and scalable monitoring solution that demonstrates Industry 4.0 principles.

By addressing these gaps, the proposed project will deliver a combined energy and process dashboard using affordable hardware, industrial communication protocols and a user-oriented interface. The next chapter outlines the detailed methodology for developing and validating the system.

3 Methodology

This chapter describes the design, implementation and validation of the developed Energy Monitoring System (EMS) for the Festo CP Lab station. The methodology combines multi domain sensing, real time data acquisition, OPC UA communication and structured experimental testing. This allows for the evaluation of electrical and pneumatic energy behaviour under different operating conditions. The main objective is to quantify system performance and assess the effectiveness of the monitoring system in capturing process related energy trends.

3.1 System architecture

The EMS was designed using a structure that consists of sensing, data acquisition, communication and visualisation elements. The overall systems data flow is shown in Figure 7 below.

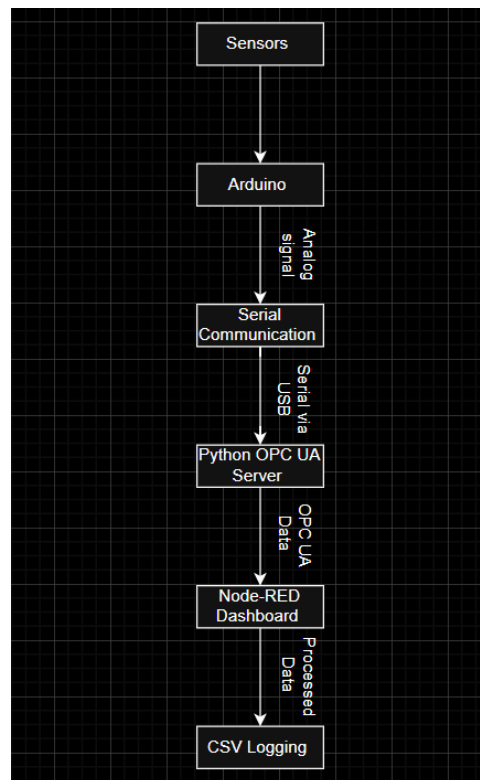


Figure 7: Shows the systems data flow.

The architecture comprises of:

- Sensors for electrical, pneumatic and environmental parameters.
- Arduino Uno for data acquisition and pre-processing.
- Python OPC UA server for data fusion and industrial communication.
- Node-RED dashboard for real time visualisation.
- CSV logging for post processing and analysis.

This structure was chosen due to its compatibility with industrial communication protocols, ability to capture real time data, storage of historical data and its modularity and scalability, as required by objectives 1 and 2. The use of OPC UA allows for interoperability with industrial systems such as the Festo CP Lab, allowing for both internal EMS data and external process data to be analysed within a unified structure.

3.2 Hardware selection and implementation

Figure 8 below shows the used Arduino and sensor configuration showing how data flows from each branch of the EMS system.

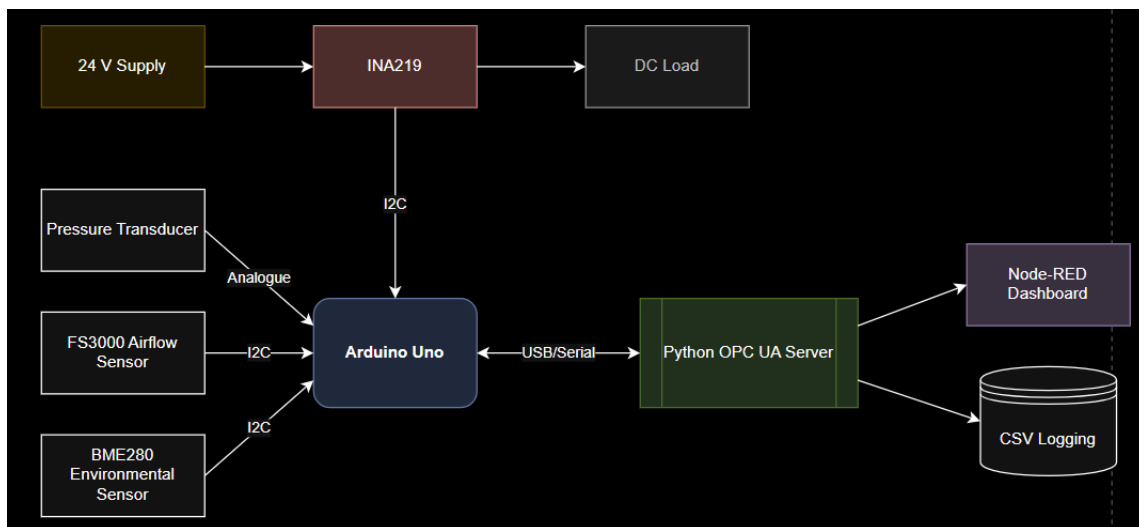


Figure 8: Shows Arduino and sensor configuration.

3.2.1 Experimental platform

The experimental platform consisted of a Festo automated production cell which has a conveyor system for pallet transportation, a robotic arm that performs pick and place operations, pneumatically actuated stoppers triggered by NFC detection and a compressed air supply regulated to approximately 6.5 bar. Each pallet represents a single mobile phone assembly cycle which allows for a direct correlation between energy consumption and production output. This setup represents realistic industrial conditions where electrical and pneumatic systems interact directly. The Festo automated production can be seen in Figure 9 below.

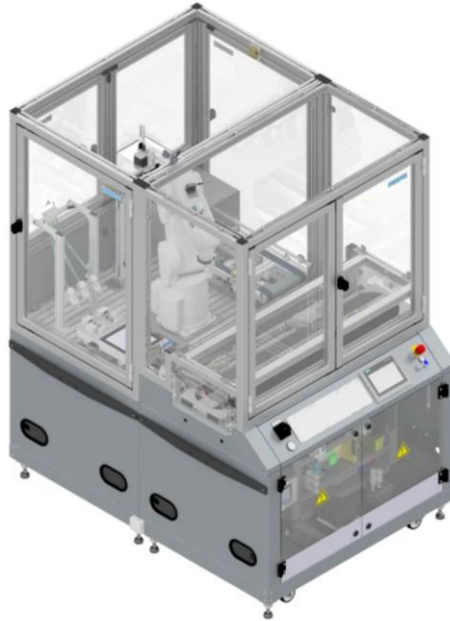


Figure 9:Shows the Festo automated production cell.

3.2.2 Sensor selection and justification

Four sensors were used to capture energy behaviour across electrical, pneumatic and environmental systems which supports objective 1.

1. Electrical measurement (INA219)

The INA219 current sensor shown below was used to measure supply voltage (V), current (mA) and electrical power (W). The sensor was connected in series with the 24V DC supply line and the conveyor motor controller.



Figure 10:Shows the INA219 current sensor module.

The INA219 was selected because of its ability to measure voltage, current, and power simultaneously. Its I²C communication helps to reduce wiring complexity. It also has a built in calibration system and high side measurement capability, making the INA219 suitable for monitoring DC loads.

2. Pneumatic supply pressure

A 100 PSI transducer was used to measure the supply air pressure as shown below. This transducer works by outputting a voltage of around 0.5 to 4.5V which is then converted to pressure (P) using the formula: $P = \frac{V - V_{min}}{V_{max} - V_{min}} \times P_{max}$. Pressure is then converted to bar for consistency with the Festo's pressure gauges and to match industrial standards.



Figure 11: Shows the 100PSI transducer.

This sensor was chosen due to its compatibility with the Arduino's analog input, being able to meet the Festo CP Lab's pneumatic ranges and its sensitivity for capturing pressure fluctuations in real time.

3. Airflow measurement (FS3000-1005)

The FS3000 sensor shown below was used to measure exhaust air velocity to detect any blockages in the pneumatic lines. A manufacturer's look up table was used with linear interpolation to convert the raw sensor values into calibrated velocity in m/s.

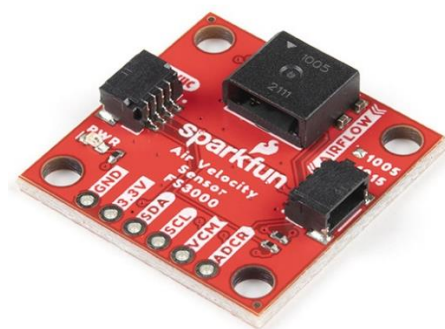


Figure 12: Shows the FS3000-1005 air velocity sensor.

The FS3000 was selected due to its ability to measure low velocities, its digital output using the I²C communication and its suitability for measuring transient pneumatic cycles due to its sensitivity.

4. Differential pressure (SDP610-500Pa)

The SDP610 was implemented to measure differential pressure across the pneumatic components used by the CP Lab as seen below.



Figure 13: Shows the SDP610-500Pa differential sensor.

The sensor range of $\pm 500 \text{ Pa}$ was insufficient for the CP Lab's high-pressure supply of around 6 bar. This led to the sensor being excluded from high pressure experiments and was reserved for future low pressure tests.

5. Environmental monitoring (BME280)

The BME280 shown below was used to measure temperature ($^{\circ}\text{C}$), atmospheric pressure (Pa) and humidity (%).

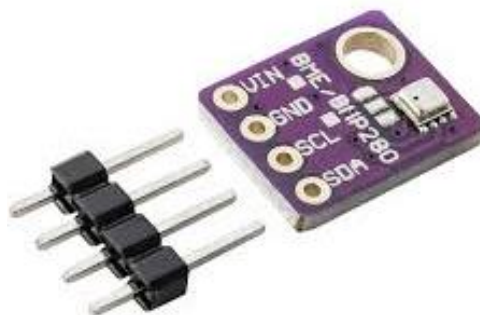


Figure 14: Shows the BME 280 environmental monitoring sensor.

Environmental monitoring ensures that external factors do not affect the testing results. It also ensures that data variability can be attributed to system behaviour rather than any ambient factors.

Table 2: Sensor specifications.

Sensor	Parameter	Range	Accuracy	Interface
INA219	Current	$\pm 3.2\text{A}$	$\pm 0.5\%$	I ² C
100 PSI	Pressure	0-6.9 bar	$\pm 1\%$	Analog
FS3000	Air velocity	0-15 m/s	$\pm 3\%$	I ² C
BME 280	Temp/Pressure/RH	-40-85 $^{\circ}\text{C}$ / $\pm 3\%$	$\pm 1 \text{ hPa}$	I ² C

3.3 Data acquisition system

3.3.1 Arduino data processing

The Arduino Uno pictured below was used as the central data acquisition unit. Its responsibilities include the following:

- Sensor initialisation
- Real time data collection
- Signal conditioning
- Unit conversion
- Transmission of data using serial

To improve the stability and reliability of the of the measurements taken, the analog pressure readings were averaged using many samples, the airflow measurements were added using look up tables and an INA219 calibration was done for a 32V and 2A range. The code used to set up the acquisition unit and sensor readings can be viewed in the appendix.

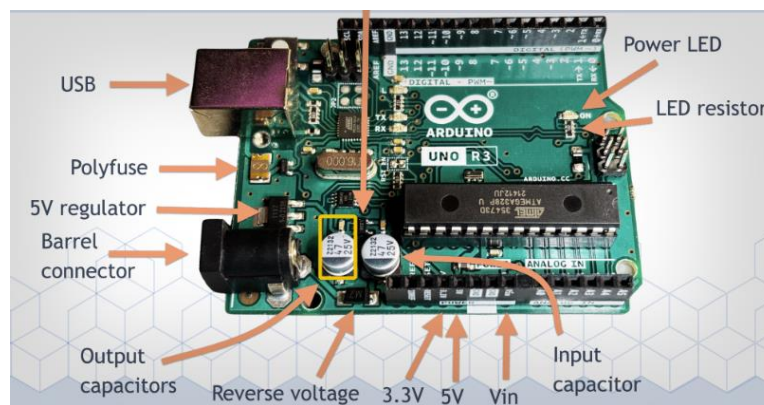


Figure 15: Shows the Arduino Uno microcontroller.

3.3.2 Sampling method

The sensor data was sampled at a rate of 10 Hz (100 ms intervals) which was used to capture transient events such as pneumatic actuation, enable sufficient temporal resolution for analysing energy and to reduce excessive noise and data redundancy. A trade off was made between sampling frequency and signal stability to ensure the measurements were reliable.

3.3.3 Serial communication

Data was transferred using serial communication at 9600 baud using a CSV format. Each data frame had electrical measurements, pneumatic pressure, airflow velocity and environmental parameters. This formatting allows for easy integration with the python data processing.

3.3.4 Post processing

Raw sensor data obtained from the data acquisition system contained noise and transient fluctuations, especially in the current measurements due to electrical switching and sensor sensitivity. To solve data quality issues and to ensure reliable analysis, a post processing filtering was applied using python which is available to view in the appendix. The filtering process consisted of the following stages:

1. Clamping of invalid values
Negative current values and unrealistic readings were removed using a lower bound threshold. This ensured that measurement noise did not introduce nonphysical values into the dataset.
2. Median filtering
A centred median filter was applied using a sliding window to remove sample spikes and outliers. This is effective for removing impulsive noise without distorting the signal.
3. Moving average filtering
A moving average filter was then used to smooth the signal and reduce high frequency noise. This resulted in a more stable representation of the systems behaviour.

The filtered current was stored as a new variable, and the electrical power was recalculated using equation 1, where the filtered values were used. This ensured that there was consistency between the current and power measurements.

The use of filtering improves the clarity of trends in the data while preserving the overall system behaviour which allows for a more accurate calculation of energy consumption and comparison between test conditions.

$$P = V \times I$$

where P = Power (W), V = Voltage (V), I = Current (A)

Equation 1: Power formula (Nilsson, J. W. and Riedel, S. A. (2019)).

3.4 OPC UA integration

A python based OPC UA server was created to receive Arduino data via serial communication to analyse and validate incoming values, to convert sensor data to OPC UA nodes and to integrate external Festo OPC UA data with the Arduino data. The python code used to create the server is available in the appendix.

- EMS nodes: AtmosphericTemperature, ConveyorCurrent_filtered, PneumaticPressure, AirVelocity etc.
- Festo node: ActivePowerL1_W (total AC power).

OPC UA was used because of its wide use in industrial automation, its platform independence and its ability to allow multiple data sources. This enables real time data comparison between the developed EMS and the Festo's built in monitoring system.

3.5 Node-RED dashboard

A real time monitoring dashboard was created to visualise the data from the EMS. The dashboard provides a user interface used to observe the Festo's behaviour during operation and allows for immediate interpretation of electrical, pneumatic and environmental parameters.

The dashboard was created using Node-RED, which allows data driven applications to be constructed using a graphical flow based programming method. Node-RED was selected due to easy integration with OPC UA and its ability to quickly develop interactive dashboards.

3.5.1 System integration architecture

The dashboard is part of a wider data acquisition and communication structure. Sensor data is first collected by the Arduino EMS and then transmitted to a python OPC UA server. The dashboard then retrieves this data using OPC UA client nodes within the Node-RED software. The data flow can be seen in Figure 7 in the following sub chapter.

This structure ensures that:

- All the sensor data is centralised within the OPC UA server.
- The dashboard operates independently of the data acquisition layer.
- The system remains modular and scalable.

The use of OPC UA allows the dashboard to integrate with both the EMS sensor data and external process data such as the inbuilt Festo power nodes, within a single interface.

3.5.2 Node-RED flow execution

The Node-RED dashboard was executed using a structured sequence which consisted of OPC UA client nodes, inject nodes and dashboard output nodes.

Data acquisition

Each parameter was linked to a specific OPC UA node by using a OpcUa-item node, which defines the NodeId and data type. These nodes were then connected to an OpcUa-client node that was configured to read data from the OPC UA server at regular intervals. Inject nodes were used to trigger data requests at a frequency of approximately 1Hz. This ensured that the dashboard remained active while avoiding excessive network traffic.

Data handling

The OPC UA client outputs the requested values as message payloads, which are then routed directly to visualisation nodes. When additional formatting was required, some simple processing was performed within Node-RED.

Flow structure

The flow was organised into similar branches that corresponded to specific parameters such as:

- Atmospheric pressure
- Atmospheric temperature
- Airflow velocity
- Electrical current and voltage
- Pneumatic pressure
- Festo AC power

This structure as shown in Figure 16 below, allows for additional variables to be added without modifying the existing flow. A full sized and more detailed diagram is available in the appendix.

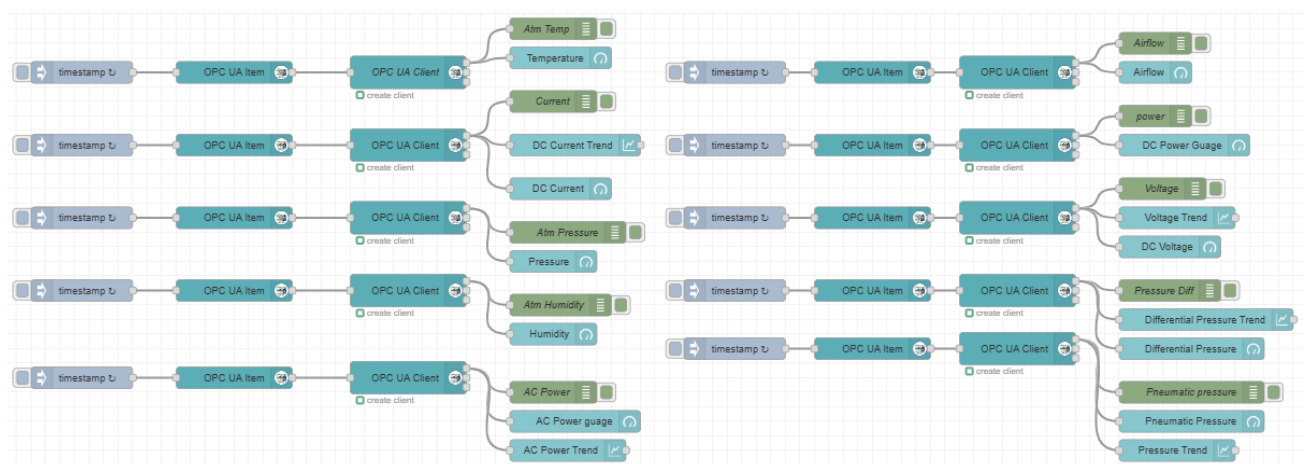


Figure 16: Shows the Node-RED flow diagram with OPC UA client, inject nodes and dashboard connections.

3.5.3 Dashboard design and visualisation

The dashboard interface was created using the Node-RED dashboard module, which provided a range of visualisation components.

Gauge visualisation

Gauge widgets were used to display real time values for key parameters such as:

- Temperature (°C)
- Pressure (bar)
- Airflow (m/s)
- Voltage (V)

- Current (mA)
- Power (W)

Each gauge was configured with appropriate ranges and colour thresholds to give an intuitive visual design.

Time series charts

The time series charts were implemented for the following dynamic variables:

- Electrical power
- Pneumatic pressure
- Airflow

These charts allow for trends over time to be observed. This is essential for identifying transient events and system behaviour during different operating conditions.

Layout design

The dashboard layout was organised into logical groups to improve usability and visibility. Related variables were grouped together such as electrical measurements together and pneumatic measurements together as seen in Figures 17 and 18 below. This allows users to quickly interpret the system's behaviour.



Figure 17: Shows the dashboard layout with real time gauges and time series plots.

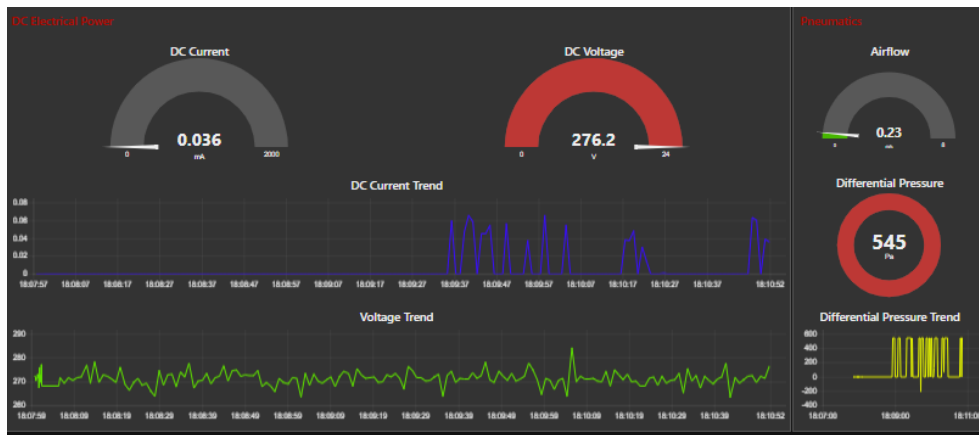


Figure 18: Shows the dashboard layout with real time gauges and time series plots.

3.5.4 Design justification

The use of Node-RED for the dashboard development has many advantages such as:

- Rapid development – the flow based development allows for fast prototyping and modification.
- OPC UA compatibility – allows for direct integration with industrial communication protocols.
- Real time visualisation – gives immediate feedback when the Festo lab is running.
- Modularity – allows for easy expansion to include additional sensors or parameters.

The dashboard also complements the CSV logging system, by providing a real time representation of the data. This is extremely useful when it comes to testing and validating the EMS.

3.5.5 Experimental testing role

The dashboard played a vital role when it came to the testing stage of the project. This is because it allowed for:

- Real time verification of sensor functionality.
- Immediate detection of any abnormal behaviour such as airflow blockages.
- Observation of system behaviour during different test scenarios.

An example of the dashboard's usefulness was seen during the blocked exhaust test, where changes in airflow were immediately visible on the dashboard.

3.5.6 Limitations

Despite the Node-RED's advantages, it also has some limitations such as:

- The data refresh rate being limited by polling frequency.
- Visualisation accuracy depending on the sensor quality.
- Node-RED can introduce some latency when displaying the data.

However, these limitations do not have a significant impact on the dashboard's usefulness as a monitoring and validation tool.

The Node-RED dashboard provides an effective real time visualisation tool for the EMS. By integrating OPC UA communication with intuitive graphical displays, the dashboard allows for monitoring, validation and interpretation of the systems behaviour. This enhances the overall functionality of the EMS and supports the analysis presented in later chapters.

3.6 Data logging and storage

All the collected data was logged in CSV format with timestamps. Each dataset included voltage, current, power, pneumatic pressure, airflow velocity, environmental parameters and the Festo AC power. This logging method allows for easy post processing, statistical analysis and creation of graphical visualisation.

3.7 Experimental procedure

Table 3 below shows the experimental procedure created to evaluate the systems performance under a set of controlled conditions. Each test was run for 60 – 600s with 10 Hz logging which produced around 600 to 6000 samples. One phone was produced per full run cycle.

Table 3: Shows developed experimental procedure.

Test No.	Experimental procedure	Purpose	Key variables observed
1	Idle test	To establish the baseline energy consumption of the system when no production is taking place.	Current, power, pressure, airflow, environmental conditions
2	Conveyor test with no load	To measure the energy consumption of the conveyor without a production load.	Conveyor current, power, time trend
3	Conveyor test with external load	To evaluate the system response to increased mechanical resistance on the conveyor.	Current increase, power increase, load response
4	Full system operation	To analyse the combined behaviour of the robotic,	Total energy, pressure, airflow, cycle behaviour

		pneumatic, and conveyor subsystems during production.	
5	Robotic arm speed variation	To investigate the effect of robot joint speed on energy consumption and cycle efficiency.	Energy per phone, cycle time, power trend
6	Pallet variation test	To assess the effect of different pallet counts on pneumatic and energy behaviour.	Pressure, airflow, energy use, cycle response
7	Blocked exhaust test	To examine how restricted airflow affects pneumatic system behaviour.	Air velocity, pressure, abnormal air flow behaviour
8	No fuses test	To compare energy consumption with and without the fuse handling stage.	Energy per phone, process time, cycle comparison

3.8 Data analysis methods

Electrical energy calculation

To calculate the electrical energy equation 2 was used.

$$E = \sum P \times \Delta t$$

where E = Total Energy(Wh), Σ = addition, P = Power(W), Δt = change in time(s)

Equation 2: Electrical energy equation.

Pressure analysis

- Mean pressure
- Pressure range
- Standard deviation
- Time below threshold

Efficiency analysis

Energy consumption was evaluated relative to the cycle time and the number of products produced.

Validation against the Festo system

The EMS data was compared with the Festo's AC power measurements to assess its consistency and reliability.

3.9 Assumptions

The following assumptions were made:

- The supply pressure stays roughly constant due to regulation
- One product is produced per cycle
- Environmental conditions stay stable
- Sensor calibration remains consistent throughout testing

3.10 Limitations

This study has limitations such as:

- Differential pressure sensor range limitations
- The INA219 only measures a single section of the system
- The Festo's power measurements represents the total systems consumption
- Noise and transient effects may affect the high frequency measurements

3.11 Summary

This methodology integrates multi domain sensing, real time communication and structured experimentation to evaluate energy consumption in an automated manufacturing system.

This approach allows for a detailed analysis of the systems behaviour and for a validation of the developed EMS.

4 Results

4.1 Results overview

The experimental method generated a set of measurements that could be used to evaluate the behaviour of the Festo CP Lab under a range of different operating conditions. The EMS measured electrical, pneumatic, airflow and environmental conditions while the station was evaluated in idle mode, conveyor only mode, overloaded conveyor mode, full production mode, pallet count variation, robot speed variation and blocked exhaust simulation. These results allowed the energy behaviour of the station to be assessed at subsystem level and at whole cell level.

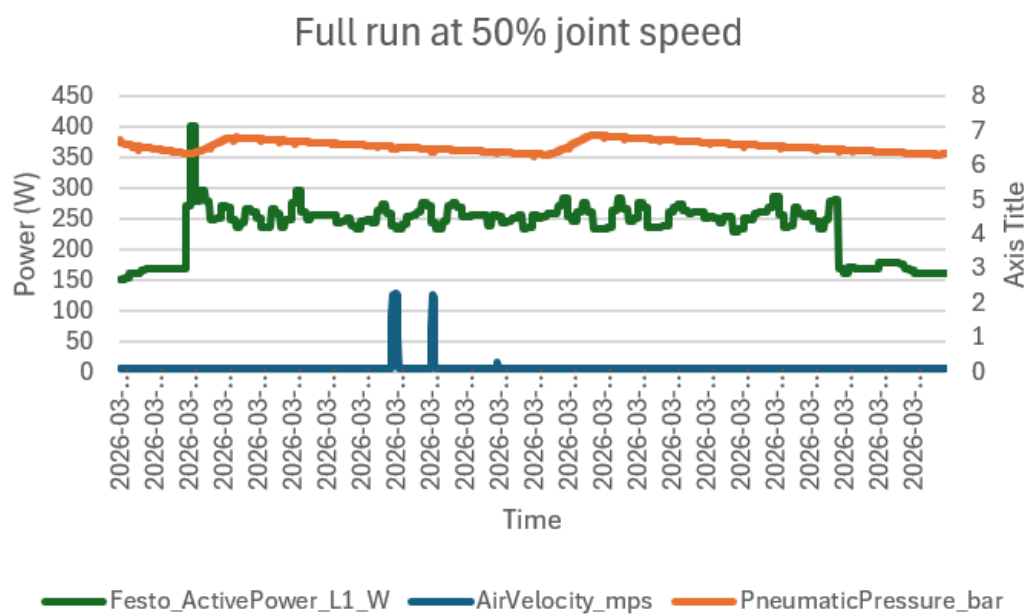


Figure 19: Show the Full production run at 50% joint speed showing AC power variation over time.

The most important outcome from the results was that the EMS provided a way to compare process modes using the same data structure. The filtered signal processing improved the quality of the current data which made the comparison between tests more reliable. This was important because the raw INA219 current data contained rapid fluctuations and non-physical values, whereas filtering made the results much easier to interpret and more suitable for analysis.

4.2 Experimental test matrix

The tests were designed to isolate the effects of different process conditions on energy consumption and system behaviour. Table 4 below shows a summary of the test conditions used in the experimental tests.

Table 4: Shows experimental test matrix.

Test ID	Test conditions	Purpose	Main variables measured
T1	Idle test	Baseline energy consumption	AC power, conveyor current, pressure, air velocity
T2	Conveyor only	Conveyor baseline under normal operation	Conveyor current, conveyor power, pressure
T3	Conveyor with force applied	Conveyor resistance test	Conveyor current, conveyor power
T4	3 conveyors running for 5 minutes	Multiload conveyor behaviour	Current, power, energy
T5	Full run 1 pallet	Full production case	Energy per phone, pressure, air velocity
T6	Full run 10 pallets	Full production case with many pallets	Energy per phone, pressure, air velocity
T7	Full run no fuses	Effects of removing fuse handling	Energy per phone, process energy
T8	Full run at 10% joint speed	Low speed robot case	Cycle time, energy per phone
T9	Full run at 25% joint speed	Intermediate robot speed	Cycle time, energy per phone
T10	Full run at 50% joint speed	Intermediate robot speed	Cycle time, energy per phone
T11	Full run at 100% joint speed	Maximum tested robot speed	Cycle time, energy per phone
T12	Blocked exhaust	Fault and blockage condition	Air velocity, pressure

This test matrix is important because it shows that the results were not collected randomly but selected to answer a specific question about the energy behaviour of the system, strengthening the validity of the experimental design.

4.3 Baseline and idle consumption

The idle test was conducted to establish the baseline energy consumption of the CP Lab station when powered but not actively producing. The baseline energy consumption is an important point because it shows how much energy the station uses when no product is being assembled. The idle data showed a mean Festo active power of approximately 158 Watts (W), which remained roughly constant throughout the test. This shows that the station has a big standing power demand even when production is not taking place.

The conveyor only test produced a similar overall AC power output to the idle test. The filtered conveyor current showed the expected motor load. The mean filtered conveyor motor power was roughly 9W which confirms the conveyor on its own contributes a small portion of the total stations power. In comparison most of the measured power came from the broader CP lab infrastructure rather than the conveyor branch alone.

This result shows that idle losses are a meaningful part of total energy usage. This means that reducing idle time could provide a big energy saving benefit than trying to make minor changes to the conveyor branch alone. The result also supports the need for a real time EMS due to a lack of monitoring making the standing energy demand hidden from the operator.

Table 5: Shows the idle and baseline results.

Test	Duration (S)	Mean AC power (W)	Peak AC Power (W)	AC energy (Wh)	Mean filtered DC power (W)	Peak filtered DC power (W)
Idle test	660	158.2	158.8	29.0	9.11	10.70
Conveyor only	110	157.0	157.5	4.80	9.23	10.61

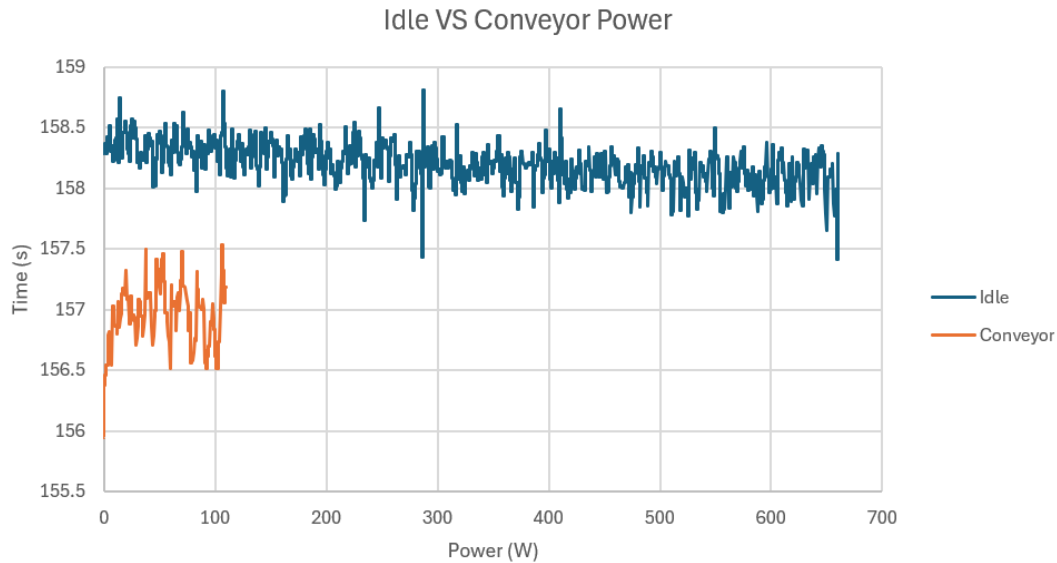


Figure 20: Shows a comparison of idle and conveyor only AC profiles.

4.4 Conveyor loading and overload behaviour

The conveyor only test was compared with the conveyor test where a manual force was applied to the moving belt. This was used to simulate added resistance and to determine if the EMS could detect change in electrical demand. Filtered current and power results showed a noticeable increase when the conveyor had added resistance. The mean filtered conveyor power increased from the normal low load to a clearly higher value under a higher force load. The peak values also greatly increased.

The results show that the INA219 is sensitive enough to detect changes in the conveyors operating condition, meaning the EMS could be used for condition monitoring or fault detection. For example, if a conveyor starts experiencing excessive resistance, the system could identify the power increase before a full failure takes place. This makes the data useful for both energy monitoring and for maintenance and diagnostics applications.

The test with three conveyors running for 5 minutes further confirmed that the EMS can distinguish different system configurations. Although the change in total power was not as significant, the results showed a consistent increase in station activity, showing that the EMS can track longer lasting load variations as well as short transient events.

Table 6: Shows conveyor load comparison.

Test condition	Mean filtered current (mA)	Std. dev. Current (mA)	Min current (mA)	Max current (mA)	Mean filtered power (W)
Conveyor only	386.9	17.5	331.1	444.9	9.23
Conveyor with added force	Higher than baseline	Higher than baseline			14.11
3 conveyors running	Lower or comparable depending on operating load				8.83

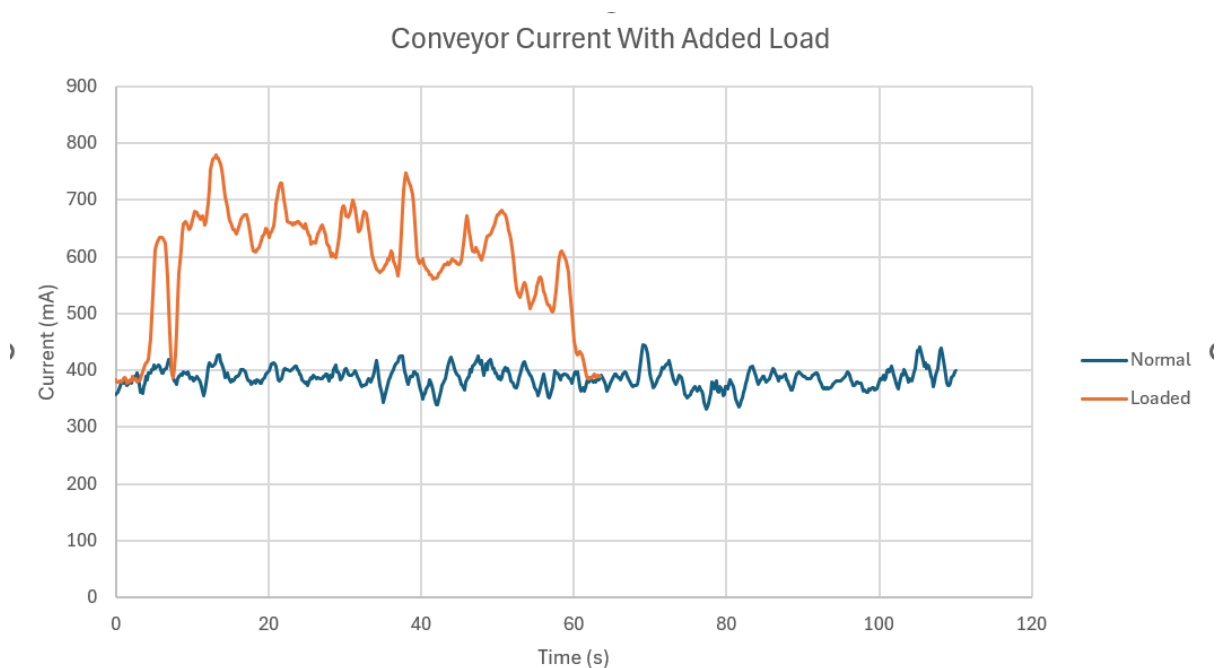


Figure 21: Conveyor current under normal operation and under additional loading.

The most important interpretation of this data is that the EMS can detect relative changes even if the absolute current values are influenced by sensor noise. This is useful as the project focuses on monitoring and comparison rather than precision instrumentation.

4.5 Pneumatic supply pressure behaviour

The pneumatic pressure values were used to assess whether the number of pallets running in the system affected the stability of the air supply. The two tests analysed were the full run

with one pallet and the full run with ten pallets. Mean pressure in both cases was quite small which showed that pallet count did not have a meaningful effect on the regulated supply pressure.

This result shows that the pneumatic regulator was performing consistently and that the pressure supply remained stable even when pallet count changed. The result also suggests that the main line pressure was not heavily affected by additional pallets moving through the conveyor system. Therefore, the results imply that any differences in overall energy usage in these tests are not caused by large pressure changes, but by cycle structure and actuation behaviour.

For the purpose of this project this result is strong because it shows that the pneumatic system is strong and stable and confirms that the EMS can measure stability quantitatively. Instead of simply stating that the pressure looked stable, the analysis shows numerical proof by using mean pressure, standard deviation, range and the amount of time spent below a certain threshold.

Table 7: Shows the pneumatic pressure comparison.

Test condition	Duration (s)	Mean pneumatic pressure (bar)	Std. dev. Pressure (bar)	Pressure range (bar)	Time below 6.35 bar (%)	Mean air velocity (m/s)
Full run (1 pallet)	114.2	6.573	0.176	0.638	7.5	0.136
Full run (10 pallets)	116.5	6.560	0.160	0.682	8.4	0.135

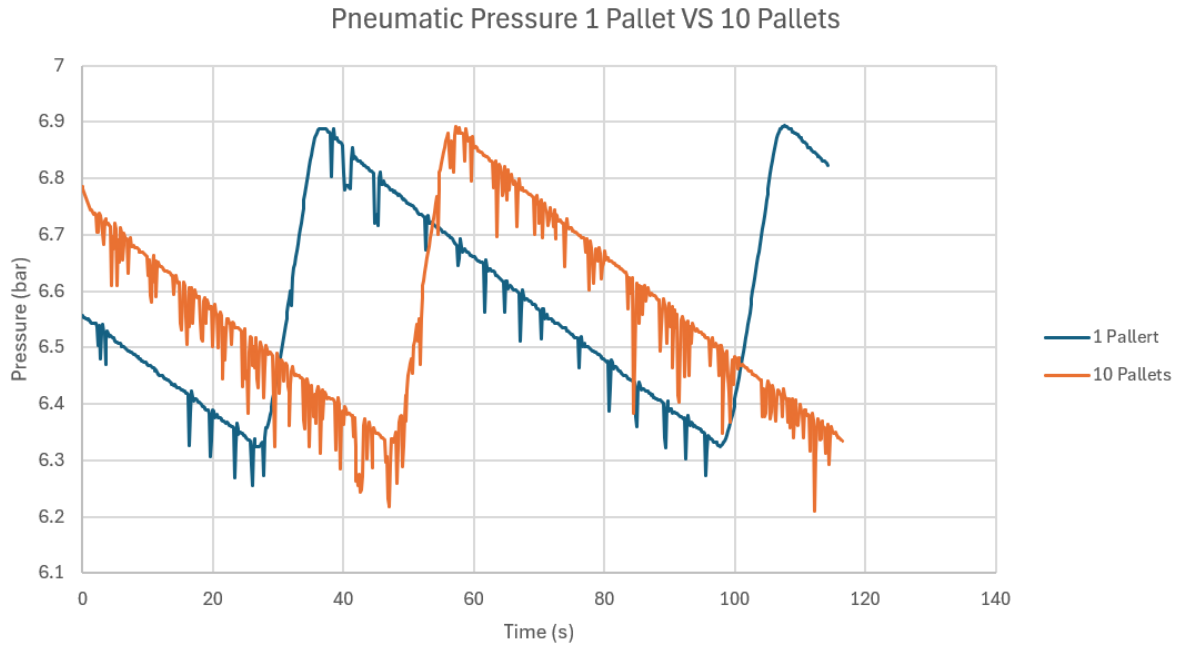


Figure 22: Shows pneumatic supply pressure during 1 pallet and 10 pallet tests.

The small difference between the two tests shows that the pneumatic supply was not significantly affected by the number of pallets running in the system. This is important as it shows that pallet count variation does not have a big pneumatic penalty under the tested operating conditions.

4.6 Exhaust airflow behaviour

The exhaust airflow results were used to assess the vacuum suction process that is associated with PCB handling in the robotic arm procedure. This was an important part of the study as the suction tooling relies on air movement, meaning that any restriction or blockage in the exhaust could affect the process reliability and pneumatic energy usage. The FS3000 sensor provided a useful way to monitor the behaviour of the exhaust line during the different operating conditions. It is important to note that this is the only process that involves the release of compressed air directly into the atmosphere, making the exhaust airflow value important for evaluating pneumatic consumption.

The results for the normal production tests showed that the mean air velocity was very similar for the 1 pallet and 10 pallet tests. In both tests the average airflow remained close to 0.135m/s. This shows that the pallet count had very little influence on the exhaust behaviour during normal operation. The pressure values were also very similar at 6.56 bar, suggesting that the regulated air supply remained stable and that the vacuum exhaust demand was determined by the suction step rather than the number of pallets moving through the system.

Table 8: Shows the exhaust airflow comparison.

Test condition	Duration (s)	Mean air velocity (m/s)	Min air velocity (m/s)	Max air velocity (m/s)	Air velocity range (m/s)	Mean pneumatic pressure (bar)	Std. dev. pressure (bar)
Full run, 1 pallet	114.2	0.136	0.100	2.490	2.390	6.573	0.176
Full run, 10 pallets	116.5	0.135	0.101	2.418	2.317	6.560	0.160
Blocked exhaust	56.0	0.201	0.102	3.088	2.986	6.697	0.152

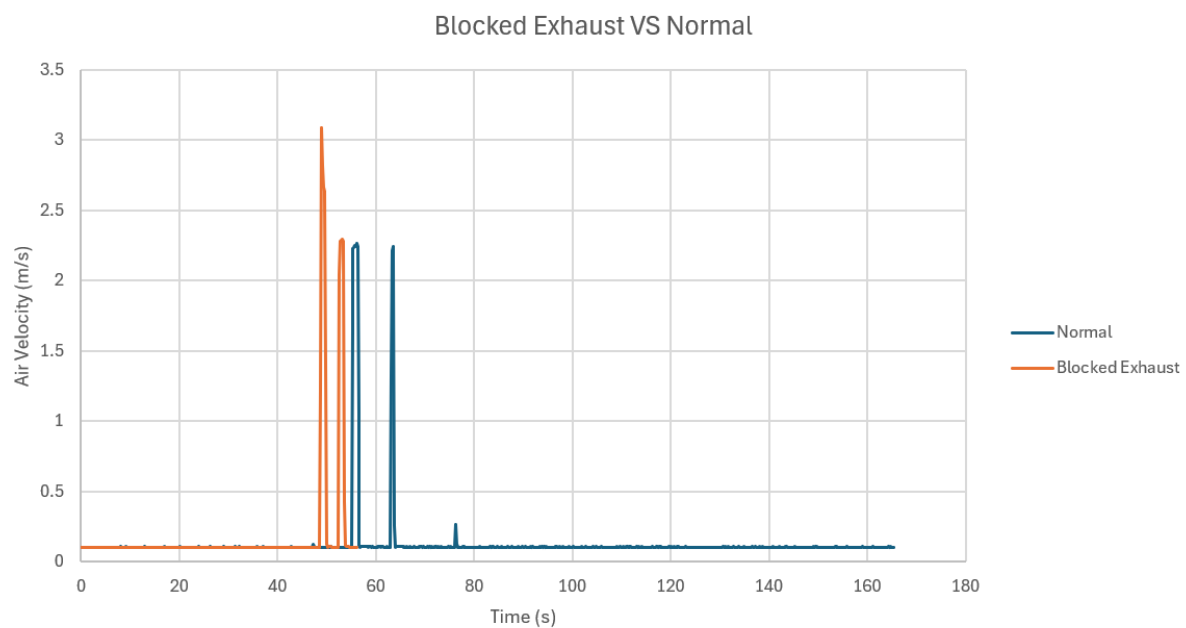


Figure 23: Shows airflow behaviour during blocked and normal tests.

The blocked exhaust test produced a noticeably different airflow profile. In this test the mean air velocity was around 0.201m/s and the maximum observed velocity rose to 3.088m/s. The airflow variation was higher which suggests a more disrupted exhaust condition. Although the pneumatic supply pressure did not drop, the higher and irregular airflow response suggests that the exhaust path was no longer behaving normally. This proves that the EMS can detect abnormal exhaust conditions and can support fault detection within the pneumatic subsystem.

The airflow and pressure results should be interpreted together. The similar mean pressure in all tests shows that the main air supply remained regulated, but the airflow data shows that the exhaust behaviour changed when the line was partially blocked. This demonstrates that that pressure variable alone may not be enough to detect faults, but combining pressure and airflow gives a better picture of the systems behaviour.

Overall, the exhaust airflow results show that the EMS was able to distinguish between normal suction and abnormal conditions. This supports the use of the airflow sensor as a monitoring device and a diagnostics tool for identifying pneumatic faults in the Festo CP lab station. The blocked exhaust test therefore strengthens the project by showing that the EMS can collect meaningful process deviations as well as standard energy data.

4.7 Full production cycle energy usage

The full production tests are the most important results from this project because they link energy consumption to product output. Since each full run produces one phone, the total energy used in each run can be treated as energy per phone, giving a clear and meaningful production key performance indicator (KPI).

The full run tests showed that energy per phone varied depending on the operating configuration. The 1 pallet and 10 pallet tests had similar energy values which suggests that changing pallet count did not cause substantial changes in the whole systems energy demand. However, the run with no fuses used less energy than the standard full run suggesting that the pneumatic actions and extra steps associated with the fuses contributed to the total energy consumption.

The most significant finding from the full production tests was the relationship between robot joint speed and energy per phone. Robot speed tests suggested that lower joint speeds increase the cycle time which also increases the energy per phone. The 10% joint speed had the highest energy consumption per phone while the 100% joint speed had the lowest energy consumption. This means that in this project experiment, faster robot joint speed was more energy efficient on a phone per production basis.

This is important because it shows that energy efficiency is not always improved by slowing down the process. Although slower motion may seem gentler and more controlled, it increased the total energy consumed for each phone because of the system remaining active for a longer period. This proves that the EMS can show that production and energy efficiency are closely linked in the CP lab station.

Table 9: Shows the full run energy comparison of the different tests.

Test condition	Duration (s)	Mean AC power (W)	Peak AC power (W)	AC energy (Wh)	Energy per phone (Wh/phone)
Full run with 1 pallet	114.2	242.4	502.5	7.70	7.70
Full run with 10 pallets	116.5	251.7	369.5	8.12	8.12
Full run with no fuses	90.2	235	368.3	5.89	5.89

Table 10: Shows robot speed comparison tests.

Joint speed	Duration (s)	Mean AC power (W)	Peak AC power (W)	AC energy (Wh)	Energy per phone (Wh/phone)	Phones per kWh
10%	633.1	235.6	705.8	41.44	41.44	24.1
25%	266.4	241.2	352.8	17.83	17.83	56.1
50%	165.4	237.4	401.5	10.89	10.89	91.8
100%	111.4	244.1	369.6	7.56	7.56	132.3

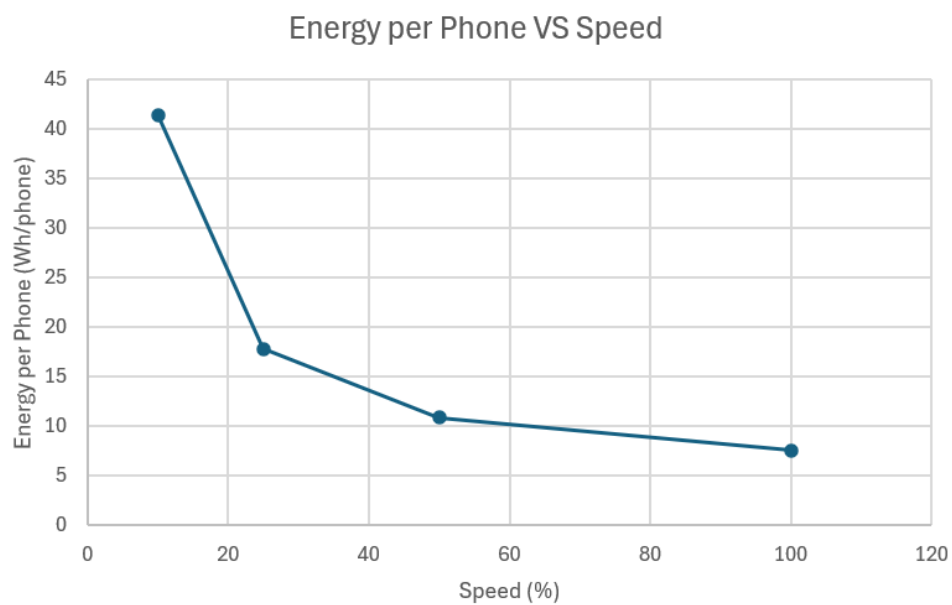


Figure 24: Energy consumption per phone as a function of robot joint speed.

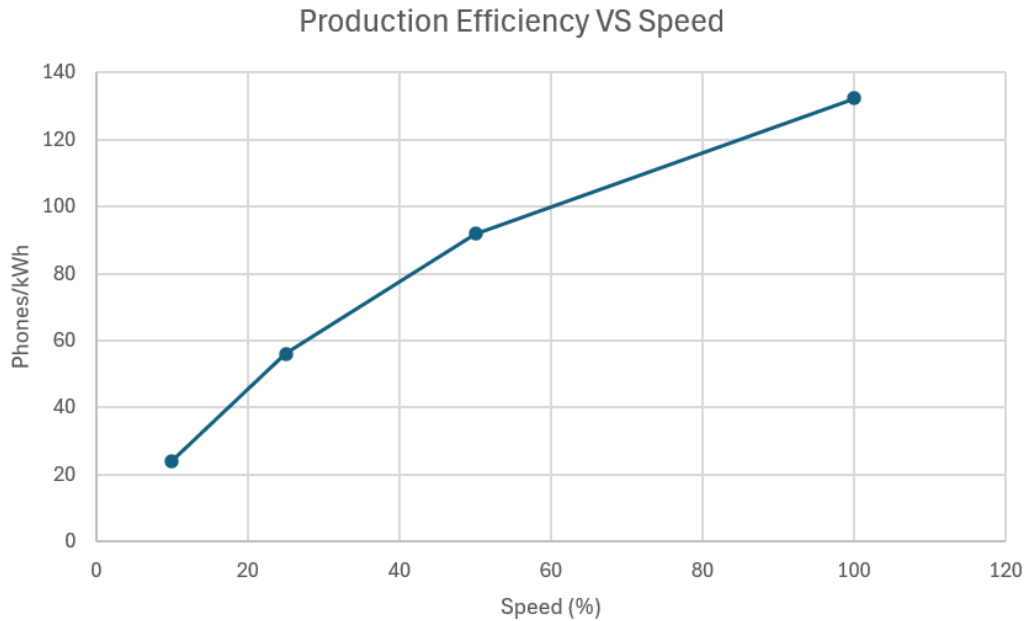


Figure 25: Production efficiency expressed as phones per kWh against robot joint speed.

The table shows that the 100% joint speed test had the best energy performance in terms of energy per phone and production efficiency. The 10% joint speed test was the least efficient because the cycle took longer even though the mean power was similar.

4.8 Effect of removing the fuses

The no fuses test was done to see if removing one of the pneumatic handling stages would have a measurable effect on the energy demand of the full production cycle. This comparison is important because it isolates the influence of a step in the process while keeping the rest of the production cycle the same. The results showed that removing the fuse handling step reduced the total energy required per phone suggesting that this step contributes the electrical demand.

The standard full run with ten pallets had a higher total energy use than the no fuses test. This means that the extra handling and actuation adds measurable energy usage. The mean AC power was slightly lower in the no fuse test which confirms that the change was not only due to a shorter cycle but also due to reduction in average system loading.

Test condition	Duration (s)	Mean AC power (W)	Peak AC power (W)	AC energy (Wh)	Energy per phone (Wh/phone)	Mean filtered DC power (W)	Mean pneumatic pressure (bar)

Full run, 10 pallets	116.5	251.7	369.5	8.12	8.12	6.95	6.560
Full run, no fuses	90.2	235.0	368.3	5.89	5.89	3.75	6.596

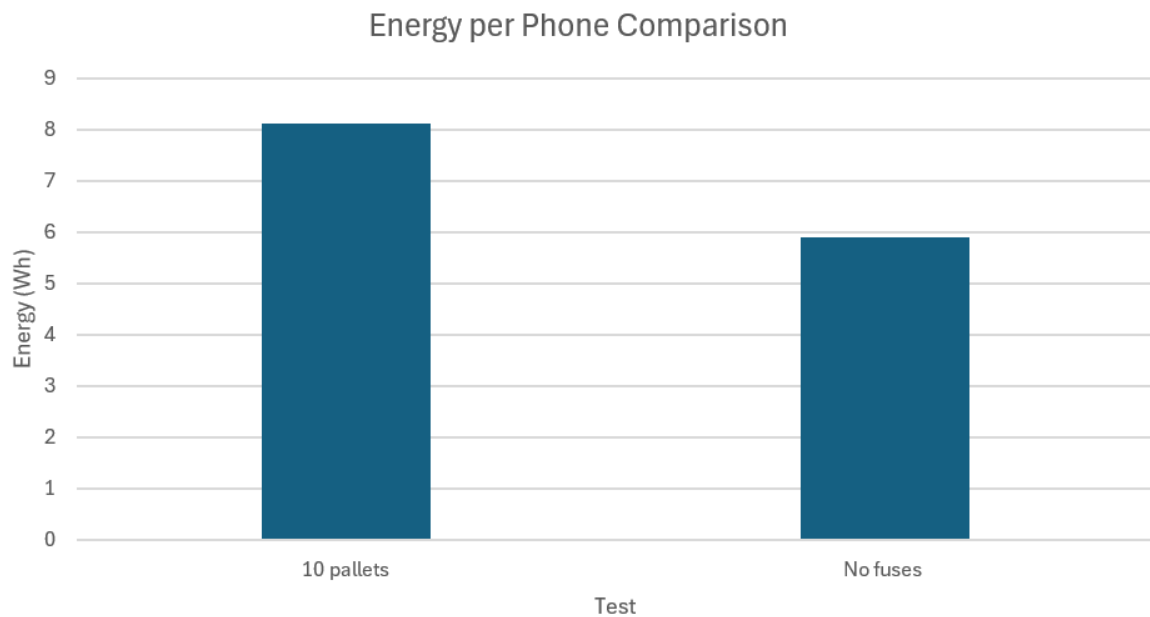


Figure 26: Shows a comparison of energy consumption per phone for normal run and nor fuses run.

The pneumatic pressure remained very similar in both tests suggesting that the main supply regulation remained stable and that the energy difference was caused by the process steps rather than changes in pressure. This shows that energy savings can be linked to reducing process steps.

This test shows that the EMS can compare two closely related configurations and can identify meaningful energy differences between them. It also shows that process changes that seem small can have a huge impact on energy efficiency which is quite important in industrial optimisation and sustainability.

Overall, the no fuses test provides evidence that the EMS can be used to evaluate process design alternatives. In industrial manufacturing environments this kind of analysis helps support decisions about handling steps being necessary or whether they introduce avoidable energy usage.

4.9 Data quality and filtering

The raw current data from the INA219 sensor was highly variable and, in some cases, contained values that were not physically realistic. To improve the interpretability of the data, the signal was filtered using clamping, median filtering and moving average, which reduced the sensor noise while preserving the basic data trend. However, it is important to note that excessive filtering can hide real short term transients and reduce the visibility of real changes. The filter settings were selected to balance noise reduction and the need to retain enough signal resolution.

The filtered results were more stable which made them suitable for comparing the test conditions. The conveyor only test showed a reduction in variation after filtering while the mean value remained significantly unchanged. This means that the filtering improved the reliability of the results without distorting the overall current demand.

This is an important aspect of the dissertation because it shows that the EMS was not just recording raw data without a data quality check being performed. The data was processed in a controlled way to improve interpretability, allowing the data to be easy to interpret and more dependable

Table 11: Shows raw results versus filtered current.

Data	Raw mean current (mA)	Raw std. dev. (mA)	Filtered mean current (mA)	Filtered std. dev. (mA)	Raw min/max (mA)	Filtered min/max (mA)
Conveyor only	389.5	41.9	386.9	17.5	286.8 / 496.5	331.1 / 444.9

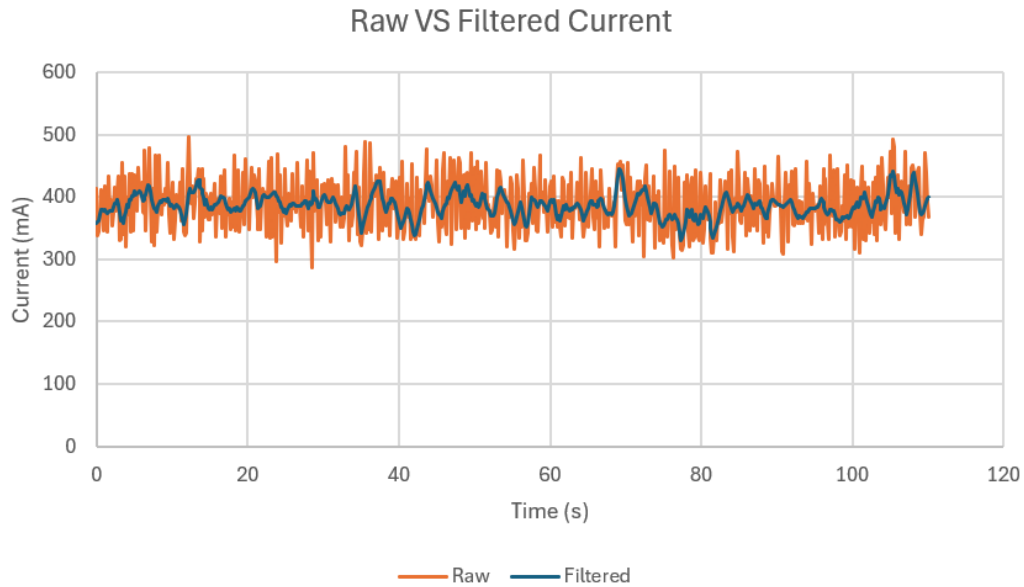


Figure 27: Shows a comparison of raw and filtered conveyor current.

Table 11 and Figure 27 above show that the filtering reduced the spread of the sensor data while keeping the mean value almost unchanged. This is the best outcome for post processing sensor data.

4.10 Summary

The results show that the EMS succeeded in monitoring the CP Lab station over a range of different operating conditions. The system managed to capture meaningful differences between the idle, conveyor only, overloaded conveyor, pallet count, blocked exhaust and speed variation tests. Filtered data was more usable than raw data and allowed the energy behaviour of the station to be analysed clearly.

The most important findings are that the idle energy consumption is substantial, the conveyor overload can be detected from current changes, pneumatic pressure remains stable when pallet count varies, airflow can identify abnormal conditions and that the robot speed has a strong influence on energy per phone. Overall, the results show that the EMS can support energy monitoring, process evaluation and fault detection within the CP lab.

5 Discussion

5.1 Overview of key findings

The experimental results show that the developed energy monitoring system (EMS) could capture important electrical, pneumatic and process related data from the FESTO CP Lab station. The system successfully identified differences between the idle, conveyor only, overloaded conveyor, normal production, blocked exhaust and different joint speed tests. These findings show that the EMS can be used not only as a monitoring tool, but also as a diagnostic and comparative analysis tool for automated manufacturing processes.

The most important findings can be summarised as follows:

- The findings indicate a significant baseline energy usage even when it was not actively producing.
- The pneumatic supply pressure remained stable across the different tests which showed good regulation but limited sensitivity for any fault detection.
- The airflow measurements showed more responsive behaviour to abnormal conditions, especially during the blocked exhaust test.
- Robot joint speed had a strong effect on the energy per phone, with faster speeds having better energy efficiency.

These outcomes support that a low cost EMS can provide valuable operational insights into automated manufacturing systems. The results also show that energy behaviour cannot be understood using electrical measurements alone, as the pneumatic and airflow data show additional process information that would have been missed.

5.2 Electrical energy behaviour

The electrical results show that the CP Lab station has a big standby energy demand. The idle test and conveyor only test produced similar AC power values with a difference of $\sim 1.2\text{W}$, which suggests that a large portion of the AC power consumption is associated with background system operations rather than production work. This background consumption comes from the controllers, sensors, pneumatic components and auxiliary electronics rather than the conveyor motors alone.

This is an important result because it highlights the inefficiencies that arise from keeping the system powered on for long periods of time without productive output. The results suggest that energy saving opportunities may exist in standby reduction, improved shutdown scheduling or better task sequencing. Reducing idle times could therefore provide a greater benefit for energy savings.

The conveyor loading tests further proved that the EMS was sensitive to changes in mechanical resistance. When a manual force was applied to the conveyor, the filtered

current and power values noticeably increased compared with the conveyor test with no resistance applied. This indicates that the current values can be used as an indicator of abnormal loading and potential mechanical stress. In industry this supports early fault detection and preventative maintenance opportunities as rising resistance can indicate wear, misalignment or obstruction.

The three conveyor test further supported that the EMS could identify changes in electrical loading across different configurations. Although the increase in power was not extreme, it was constant and measurable. This suggests that the system is suitable for comparing relative energy demand across different modes even though it is not yet intended to perform as a precision metering device.

5.3 Pneumatic system behaviour

The pressure data showed that the pneumatic supply remained regulated during the tests. The mean pressure values for the one pallet and ten pallet tests were very similar with relatively small variation. This indicates that the supply line was stable and that the regulator maintained the set pressure effectively even when the process setup changed.

From an engineering point of view, this is a positive result because it confirms that the pneumatic system functioned reliably. It also showed that the supply pressure alone is not a highly sensitive indicator of changes in production loads. In other words, pressure stability is useful for verifying correct operating pressure but does not reveal much about whether the system is working efficiently or abnormally.

The airflow results provided a better informative view of pneumatic behaviour. Under normal operation the air velocity measurements were similar for the one pallet and ten pallet tests. This suggests that the pallet count did not significantly affect the exhaust demand. The blocked exhaust test produced a more distributed airflow pattern with a higher mean velocity and a larger maximum value showing that the exhaust path had been altered in a way that changed the flow behaviour.

This comparison showed the importance of measuring more than one pneumatic variable. Pressure alone does not reveal a blocked exhaust condition, but the air velocity provided a much better indication of abnormal behaviour. This makes airflow a useful variable for fault detection and supports multi sensor monitoring in pneumatic systems.

5.4 Effects of process changes

The operating condition comparisons show that different process settings affect the EMS outputs in different ways. The number of pallets running in the system had a limited influence on pressure and airflow which suggests that the regulated supply pressure is independent of

pallet count under the tested conditions. This is consistent with the role of the pressure regulator which is to maintain a stable supply regardless of changes in demand.

The robot speed tests produced the most important performance trend in the study. As joint speed increased, the cycle time reduced and the energy consumed per phone also reduced. This shows that faster operation was more energy efficient. The reason for this is that the total energy consumed during a cycle is influenced strongly by the duration meaning that reducing the time spent powered on reduces cumulative energy demand.

This is an important finding because it shows that slowing a process down does not necessarily improve energy efficiency. In this test, lower speed increased the time in which the system remained active which outweighed potential reduction in instantaneous loads. The result therefore provides a clear output energy trade off where faster production gives better energy efficiency per unit output.

The no fuses test also showed that changes to the process sequence can affect energy demand. Removing the fuse handling step reduced the total energy per phone from 8.12Wh to 5.89Wh, showing that the pneumatic and handling actions contributed to the overall consumption. This suggests that even a relatively small change in process steps can have a measurable effect on energy use. This is valuable because it shows that the EMS can be used to compare alternative process routes and identify opportunities for simplification.

5.5 EMS performance evaluation

The EMS performed well as an integrated monitoring platform. It was able to collect data from multiple sensors, transmit the information through the communication structure and present it in a form that is suitable for analysis. This is important because many low cost monitoring systems are limited to a single type of measurement, whereas this project combined electrical, pneumatic and environmental data in one framework.

The system was particularly effective at identifying relative changes in process behaviour. It successfully distinguished between idle and active operation, normal and abnormal conveyor conditions, stable and blocked exhaust behaviour and different robot speed settings. These comparisons confirm that the EMS provides useful insights rather than simple raw data logging.

The OPC UA integration also adds value because it makes the system more compatible with industrial communication standards. This means the monitoring framework is not limited to the lab setup and could be extended to other automates systems. The Node-RED dashboard improved usability by allowing real time visualisation of key variables during testing.

The results also show that the system is dependent on the quality of its sensors and preprocessing. The INA219 measurements required filtering because the raw signal contained noise and occasional non-physical values. This is not unusual in measuring systems but means that data processing is necessary before final conclusions. The filtering step made the power trends more reliable for comparison.

5.6 Data quality and signal processing

The signal processing method used had a visible impact on the quality of the results. The raw current measurements contained fluctuations that would have made direct interpretation difficult and misleading. By applying clamping, median filtering and moving average smoothing, the dataset became more reliable for analysis.

This is important because the accuracy of the power and energy calculations depends directly on the quality of the current signal. Small measurement errors can accumulate over time and affect the energy per cycle calculated. The filtering therefore improved the appearance of the plots and the credibility of the quantitative results.

The use of the filtered data also improved the strength of the project argument. It shows that the analysis was not based on unprocessed output alone but on a structured approach to data preparation. That makes the findings more robust and demonstrates awareness of practical measurement limitations.

5.7 Industrial implications

The findings have several implications for industrial energy management. The first is that idle consumption is big and should be addressed. In an industrial setting background energy demand can represent a major hidden cost, especially if the equipment remains powered for long periods of time without productive work. This suggests that scheduling, standby control and shutdown strategies could all offer energy savings.

The second implication is the importance of multi variable monitoring. The pneumatic pressure data alone would not have been enough to identify all the tested components, whereas the addition of airflow measurements improved fault detection capabilities. This supports sensor fusion in industrial diagnostics.

The third implication is that process speeds can affect energy efficiency more than people expect. In this study, increasing robot speed improved energy performance per phone because it reduced the cycle time. This shows that energy optimisation should be considered alongside productivity rather than isolation from it.

Lastly, the EMS framework itself has potential value beyond this project. A similar method could be used for monitoring other automated stations where electrical, pneumatic and environmental variables all contribute to the system behaviour. The combination of low cost hardware, industrial communication and real time visualisation makes the approach practical for industrial use.

5.8 Limitations of the study

Several limitations should be acknowledged when interpreting the results. The first is that the study was conducted on a laboratory scale CP Lab station which is less complex than a full industrial production line. Although the system provides useful insight, the results may not capture all the variability present in real manufacturing environments.

The second limitation is related to measurement resolution. The Arduino based acquisition is suitable for general monitoring but is not equivalent to high end industrial instrumentation. This means very fast transients may be under sampled or smoothed out by the filtering process.

The third limitation is about the available pressure sensing range. Some sensor options were not suitable for the high pressure parts of the system, which restricted the level of detail obtained in the pneumatic analysis. Future work could address this by using industrial grade sensors or by adding dedicated flow sensors.

The final limitation is that the analysis the analysis focused on a specific set of conditions. While the tests were enough to show the EMS capabilities, broader testing across more production states would provide a more complete assessment of performance and robustness.

5.9 Summary

The discussion shows that the EMS can provide meaningful insights into the energy and process behaviour of the Festo CP Lab. The results confirm that the standby energy consumption is substantial, pneumatic pressure is stable but not so diagnostic, airflow is better for identifying abnormalities and robot speed has a big influence on energy per phone.

The study therefore supports that multi sensor monitoring is a valuable tool for understanding and improving automated manufacturing systems. The system is useful for energy monitoring, identification of process conditions, comparing process alternatives and supporting future optimisation.

6 Conclusions and future work

6.1 Conclusions

This project set out to develop and evaluate an energy monitoring system for an automated Festo CP Lab manufacturing station. The project successfully achieved this aim by integrating multiple sensors, communication through an industrial used protocol and real time visualisation into a single monitoring framework. The resulting system was able to collect and interpret electrical, pneumatic and environmental data across different operating conditions.

The experimental results showed that the CP Lab station has a significant standby energy usage even when idle. This means that a large portion of the system's energy use is associated with maintaining readiness rather than with direct production. Reducing idle periods and using better standby control would likely improve efficiency.

The pneumatic analysis showed that supply remained stable across the tests. This shows that the regulator was functioning effectively but also means that pressure alone is not enough for detecting system behaviour changes. Airflow values proved informative for identifying abnormal exhaust conditions especially in the blocked exhaust test.

Robot speed had a big effect on energy efficiency. Within the tested range, faster motion reduced energy per phone because it shortened the cycle time and reduced the duration over which standby power was used. This demonstrates that energy efficiency and productivity are closely linked in this system.

Overall, the EMS proved to be an effective monitoring tool. It can be used to identify standby consumption, detect changes in load, compare operating modes and support process evaluation. The project therefore demonstrates that a relatively low cost monitoring system can give useful insights into the performance of an automated manufacturing cell.

6.2 Main contributions

The project makes many contributions such as:

- Developing a multi sensor EMS that is suitable for an automated manufacturing system.
- It combined electrical, pneumatic and environmental measurements within one framework
- It demonstrated real time data communication and visualisation.
- Showed that the system can detect differences between normal and abnormal operating conditions.
- Identified key factors that affect energy consumption, especially idle load and robot speed.

The contributions show that the project has value as a technical monitoring device and as a study in manufacturing energy analysis.

6.3 Recommended further work

Many improvements could be made to improve the capability of the system in future work such as:

1. Using higher precision sensors such as a more suitable pressure and flow measurement setup to improve the pneumatic analysis and to allow for a more detailed analysis of air consumption. This would make it easier to quantify pneumatic losses more accurately rather than using relative trends.
2. The data acquisition system could be upgraded to improve sampling resolution. A higher speed data logger such as a Raspberry Pi would allow for better accuracy and reduce the need for heavy data smoothing when capturing fast transients. This would improve data quality and interpretation.
3. The EMS could be extended by using automated fault detection. Currently the system is useful for observing trends and identifying faults manually, but further work could apply rule-based logic or machine learning to detect faults automatically. This would make the system more useful in real time industrial environments.
4. The monitoring system could be linked to a digital twin. Running a simulation model alongside the physical would allow for measured data to be compared with predicted system behaviour. This could support predictive maintenance, anomaly detection and process optimisation.
5. The system could be tested on a more complex industrial platform. This would help to validate how well the system scales beyond the lab environment and whether the same monitoring elements remain effective under higher loads and greater variability.
6. The energy analysis could be expanded to include carbon accounting or cost modelling. This would make the results directly relevant to sustainability and industrial decision-making by translating measured energy values into environmental and financial terms.
7. Evaluate and optimise the dashboard design through a structured user feedback and participant survey to ensure usability and operator effectiveness across a range of different skill levels.

6.4 Final remarks

The work presented in this dissertation shows that energy monitoring can be integrated successfully into an automated manufacturing environment using accessible hardware and industrial communication tools. The system developed provides a practical basis for future research into energy optimisation, condition monitoring and intelligent manufacturing.

7 References

Introspective on Achieving Information Integration and Interoperability in Process Automation.

Adebowale, J. (2026) *INTELLIGENT PERFORMANCE DASHBOARDS: REAL-TIME PROJECT TRACKING USING MACHINE LEARNING AND PREDICTIVE ANALYTICS.*

Frigerio, N., Matta, A. and Rasella, M. (2022) 'Energy monitoring of manufacturing plants: A real case application', *Procedia CIRP*, 105, pp. 770–775. Available at: <https://doi.org/10.1016/j.procir.2022.02.128>.

He, Y., Wu, P., Wang, Y., Tao, F. and Hon, B.K.K. (2020) 'An OPC UA based framework for predicting energy consumption of machine tools', *Procedia CIRP*, 90, pp. 568–572. Available at: <https://doi.org/10.1016/j.procir.2020.02.133>.

Jagtap, S., Rahimifard, S. and Duong, L.N.K. (2019) *Real-time data collection to improve energy efficiency: A case study of food manufacturer* Wiley.

Krutwig, M.C. (2019) *Suitability of OPC UA for distributed energy monitoring* Walter de Gruyter GmbH.

Lee, J. and Lee, D.Y. (2025) 'Development of energy consumption monitoring system for machine tools in accordance with ISO14955', *Manufacturing Letters*, 44, pp. 18–23. Available at: <https://doi.org/10.1016/j.mfglet.2025.06.006>.

Massaro, A. and Starace, G. (2022) *Advanced and Complex Energy Systems Monitoring and Control: A Review on Available Technologies and Their Application Criteria* MDPI AG.

May, G., Stahl, B., Taisch, M. and Kiritsis, D. (2017a) 'Energy management in manufacturing: From literature review to a conceptual framework', *Journal of Cleaner Production*, 167, pp. 1464–1489. Available at: <https://doi.org/10.1016/j.jclepro.2016.10.191>.

May, G., Stahl, B., Taisch, M. and Kiritsis, D. (2017b) 'Energy management in manufacturing: From literature review to a conceptual framework', *Journal of Cleaner Production*, 167, pp. 1464–1489. Available at: <https://doi.org/10.1016/j.jclepro.2016.10.191>.

Antus, W., Vaasa, Energy, S., Shafiekhah, M. and Laaksonen, H. *Plan a dashboard for energy measuring, improve overview of energy consumption, and increase energy recovery.*

Dussault, R. and Electric, S. *Dashboard Demystified: Making Sense of Industrial Energy Management Visualization Systems.*

Hernández, T., Lehoux, N. and Gosselin, L. *Development of an energy monitoring and control interface for the forest products industry.*

Frigerio, N., Matta, A. and Rasella, M. (2022) 'Energy monitoring of manufacturing plants: A real case application', *Procedia CIRP*, 105, pp. 770–775. Available at: <https://doi.org/10.1016/j.procir.2022.02.128>.

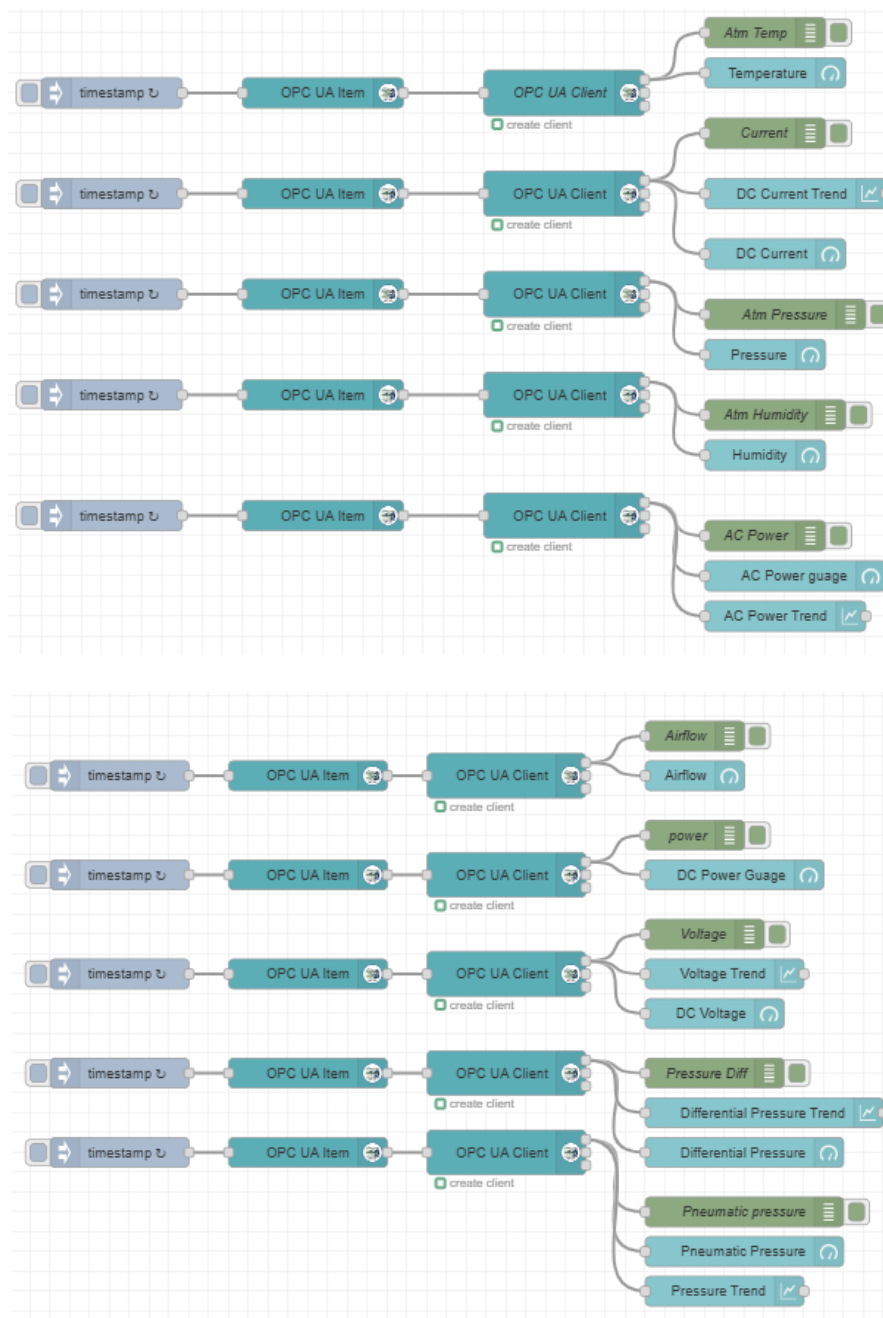
Jagtap, S., Rahimifard, S. and Duong, L.N.K. (2019) *Real-time data collection to improve energy efficiency: A case study of food manufacturer* Wiley.

Massaro, A. and Starace, G. (2022) *Advanced and Complex Energy Systems Monitoring and Control: A Review on Available Technologies and Their Application Criteria* MDPI AG.

- Uhlmann, E., Polte, J. and Geisert, C. (2025) 'IoT-based energy monitoring in production', *Measurement: Sensors*, 38, pp. 101591. Available at: <https://doi.org/10.1016/j.measen.2024.101591>.
- Busboom, A. (2024) 'Automated generation of OPC UA information models — A review and outlook', *Journal of Industrial Information Integration*, 39, pp. 100602. Available at: <https://doi.org/10.1016/j.jii.2024.100602>.
- He, Y., Wu, P., Wang, Y., Tao, F. and Hon, B.K.K. (2020) 'An OPC UA based framework for predicting energy consumption of machine tools', *Procedia CIRP*, 90, pp. 568–572. Available at: <https://doi.org/10.1016/j.procir.2020.02.133>.
- Krutwig, M.C. (2019) *Suitability of OPC UA for distributed energy monitoring* Walter de Gruyter GmbH.
- Mittal, A., Gupta, P., Kumar, V., Al Owad, A., Mahlawat, S. and Singh, S. (2023) 'The performance improvement analysis using six sigma DMAIC methodology: A case study on indian manufacturing company', *Heliyon*, 9(3), pp. e14625. Available at: <https://doi.org/10.1016/j.heliyon.2023.e14625>.
- Hetherington, V. (no date) 'The Dashboard Demystified', Dashboard Insight. Available at: dashboardinsight.com (Accessed: [Insert Date You Viewed It]).
- Pertijs, Michiel. (2014). Calibration and Self-Calibration of Smart Sensors. Smart Sensor Systems: Emerging Technologies and Applications. 17-41. 10.1002/9781118701508.ch2.
- Maag, Balz & Zhou, Zimu & Thiele, Lothar. (2018). A Survey on Sensor Calibration in Air Pollution Monitoring Deployments. IEEE Internet of Things Journal. PP. 1-1. 10.1109/JIOT.2018.2853660.
- Adewumi, A. et al. (2025) 'The Evolution of Dashboard Design: Best Practices for Data Visualization in Decision Support Systems', *Computational Engineering Journal*, 1(1).
- Nilsson, J. W. and Riedel, S. A. (2019) *Electric Circuits*. 11th edn. Pearson.

8 Appendix

8.1 Node-RED flow diagrams



8.2 Arduino code

```
#include <SparkFun_FS3000_Arduino_Library.h>
#include <Wire.h>
#include <Adafruit_BME280.h>
#include <Adafruit_INA219.h>

// SENSOR OBJECTS
Adafruit_BME280 bme; // BME280 for temperature humidity and
atmospheric pressure
```

```

Adafruit_INA219 ina219(0x41); // INA219 for DC voltage current and power
FS3000 fs; // FS3000 airflow sensor

// I2C ADDRESSES
#define BME_ADDR 0x76
#define SDP610_ADDR 0x40

// FS3000 LOOKUP TABLE
// The tables converts the manufacturers raw output points into calibrated air
velocity.
const int fsRawPoints[15] = {
    409, 915, 1522, 2147, 2752,
    3367, 3986, 4603, 5222, 5840,
    6458, 7075, 7693, 8311, 8929
};

const float fsVelocityPoints[15] = {
    0.0, 0.5, 1.0, 1.5, 2.0,
    2.5, 3.0, 3.5, 4.0, 4.5,
    5.0, 5.5, 6.0, 6.5, 7.23
};

// FS3000 RAW TO VELOCITY CONVERSION
// This converts the raw FS3000 reading into air velocity (m/s)
float lookupVelocity(uint16_t raw) {
    // If raw value is below lowest point clamp to minimum velocity
    if (raw <= fsRawPoints[0]) return fsVelocityPoints[0];

    // If raw value is above highest point clamp to maximum velocity
    if (raw >= fsRawPoints[14]) return fsVelocityPoints[14];

    // Interpolate between the two nearest points
    for (int i = 0; i < 14; i++) {
        if (raw >= fsRawPoints[i] && raw <= fsRawPoints[i + 1]) {
            float fraction = (float)(raw - fsRawPoints[i]) /
                (fsRawPoints[i + 1] - fsRawPoints[i]);
            return fsVelocityPoints[i] +
                fraction * (fsVelocityPoints[i + 1] - fsVelocityPoints[i]);
        }
    }

    //Return if no range match
    return 0.0;
}

// SDP610 CHECK
// This calculates the CRC-8 checksum so that received data is verified.
uint8_t crc8_sensirion(const uint8_t data[2]) {

```

```

uint8_t crc = 0x00;
for (int i = 0; i < 2; i++) {
    crc ^= data[i];
    for (int b = 0; b < 8; b++) {
        if (crc & 0x80) crc = (crc << 1) ^ 0x31;
        else crc <<= 1;
    }
}
return crc;
}

// SDP610 RESET
bool sdp610_softReset() {
    Wire.beginTransmission(SDP610_ADDR);
    Wire.write(0xFE);
    return (Wire.endTransmission() == 0);
}

// SDP610 DIFFERENTIAL PRESSURE READ
bool sdp610_readDiffPressurePa(float &diffPa) {

    // Request a measurement from the sensor
    Wire.beginTransmission(SDP610_ADDR);
    Wire.write(0xF1);
    if (Wire.endTransmission() != 0) return false;

    uint8_t n = Wire.requestFrom(SDP610_ADDR, (uint8_t)3);
    if (n != 3) return false;

    uint8_t msb = Wire.read();
    uint8_t lsb = Wire.read();
    uint8_t crc = Wire.read();

    uint8_t data[2] = {msb, lsb};
    if (crc8_sensirion(data) != crc) return false;

    int16_t raw = (int16_t)((msb << 8) | lsb);

    // Convert raw SDP610 output into Pascals
    diffPa = (float)raw / 60.0f;
    return true;
}

void setup() {
    // Start serial communication for output to Python
    Serial.begin(9600);
    delay(2000); // for serial monitor and sensors to stabilise
}

```

```

// Start I2C bus
Wire.begin();
Wire.setClock(100000); // normal I2C speed

Serial.println("EMS Sensors Starting...");

// BME280
// Used for environmental temperature, atmospheric pressure and humidity
Serial.print("Initializing BME280...");
if (!bme.begin(BME_ADDR)) {
    Serial.println(" FAILED");
    while (1); // Stop if sensor is not detected
}
Serial.println(" OK");

// INITIALISE INA219
// Used for voltage, current and power
Serial.print("Initializing INA219...");
if (!ina219.begin()) {
    Serial.println(" FAILED");
    while (1); // Stop if sensor is not detected
}
ina219.setCalibration_32V_2A();
Serial.println(" OK");

// INITIALISE FS3000
// Used for airflow
Serial.print("Initializing FS3000...");
if (!fs.begin()) {
    Serial.println(" FAILED");
    while (1);
}
Serial.println(" OK");

// INITIALISE SDP610
// Used for differential pressure
Serial.print("Initializing SDP610...");
if (!sdp610_softReset()) {
    Serial.println(" FAILED");
    while (1);
}
delay(50); // delay for stable operation
Serial.println(" OK");

// Dummy read to discard first pressure reading
float dummy;
sdp610_readDiffPressurePa(dummy);

```

```

    // Print CSV header so Python logging software knows the column order
    Serial.println("TempC,AbsPressurePa,AirVelocity_ms,VoltageV,Current_mA,Power
_W,Humidity_pct,DiffPressure_Pa");
}

void loop() {
    // READ BME280
    // Measure ambient environmental condition
    float temperatureC    = bme.readTemperature();
    float absPressurePa   = bme.readPressure();
    float humidityPct     = bme.readHumidity();

    // READ FS3000
    // Read airflow sensor value and convert to m/s
    uint16_t rawFS        = fs.readRaw();
    float airVelocity_ms  = lookupVelocity(rawFS);

    // READ INA219
    float voltageV        = ina219.getBusVoltage_V();
    float current_mA      = ina219.getCurrent_mA();
    float power_W         = ina219.getPower_mW() / 1000.0f;    // Convert mW to W

    // READ SDP610
    // Measure differential pressure if failed return NaN
    float diffPressurePa;
    bool ok = sdp610_readDiffPressurePa(diffPressurePa);
    if (!ok) diffPressurePa = NAN;

    // SERIAL CSV OUTPUT
    // Output all sensor values in one CSV line
    // This format is used by the Python OPC UA logging system
    Serial.print(temperatureC, 2);    Serial.print(",");
    Serial.print(absPressurePa, 2);    Serial.print(",");
    Serial.print(airVelocity_ms, 3);   Serial.print(",");
    Serial.print(voltageV, 2);         Serial.print(",");
    Serial.print(current_mA, 2);       Serial.print(",");
    Serial.print(power_W, 3);          Serial.print(",");
    Serial.print(humidityPct, 2);      Serial.print(",");
    Serial.println(diffPressurePa, 2);

    // Sampling delay 100 ms = 10 Hz sampling rate
    delay(100);
}

```

8.3 Python code

```

import sys
import time

```

```

import serial
import csv
import os
from datetime import datetime
from opcua import ua, Server, Client

# SERIAL COMMUNICATION
# COM port and baud rate used to receive data from the Arduino
PORT = "COM5"
BAUD = 9600

# CSV LOGGING SETTINGS
# Folder where logged data files will be saved
CSV_FOLDER = r"C:\Users\sheuz\OneDrive\Desktop\OPC Data"
os.makedirs(CSV_FOLDER, exist_ok=True)

# Create a new timestamped filename for each run
csv_filename = os.path.join(
    CSV_FOLDER,
    f"ems_log_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
)

LOG_EVERY_N_ROWS = 10

# FESTO OPC UA CONNECTION SETTINGS
# External Festo OPC UA endpoint and power node used for comparison
FESTO_ENDPOINT = "opc.tcp://172.21.0.60:4840"
FESTO_NODE_ACTIVE_POWER_L1 = "ns=4;s=|var|CPX-E-CEC-
C1.Application.GVL.PowerMeter.ActivePowerL1"

# polling the Festo power node in seconds
FESTO_POLL_SECONDS = 0.2

# CONNECT TO ARDUINO
# Open the serial connection to the Arduino so live sensor data can be read
try:
    arduino = serial.Serial(PORT, BAUD, timeout=2)
    time.sleep(2) # Arduino reset after serial port opens
    print(f"Connected to Arduino on {PORT}")
except Exception as e:
    print("Error connecting to Arduino:", e)
    sys.exit(1)

# PARSE INCOMING ARDUINO CSV DATA
def parse_9_float_csv(line: str):
    """
    Expected Arduino line format:
    TempC,AbsPressurePa,AirVelocity_ms,VoltageV,Current_mA,Power_W,

```

```

Humidity_pct,DiffPressure_Pa,PneuPressure_bar

Returns:
    tuple of 9 floats if parsing is successful
    None if line is invalid, incomplete, header text, or contains an error
"""
# Ignore blank lines or CSV header line
if not line or line.lower().startswith("tempc"):
    return None

# Ignore lines containing invalid values or sensor error text
if "nan" in line.lower() or "err" in line.lower():
    return None

# Split incoming CSV line into fields
parts = [p.strip() for p in line.split(",")]

# number of fields
if len(parts) != 9:
    return None

# Convert all fields into floats
try:
    return tuple(float(p) for p in parts)
except ValueError:
    return None

# FESTO EXTENSION OBJECTS
def resolve_numeric_child_node(struct_node):
    """
    Some Festo OPC UA nodes return ExtensionObjects rather than direct
    numbers.
    This function searches child nodes until a numeric child value is found.

    Returns:
        child node containing numeric value
        None if no numeric child could be located
    """
    try:
        children = struct_node.get_children()
    except Exception:
        return None

    for ch in children:
        try:
            v = ch.get_value()
            if isinstance(v, (int, float)):
                return ch

```

```

        except Exception:
            pass

    return None

# CONNECT TO FESTO OPC UA SERVER
def connect_festo(endpoint, nodeid):
    """
    Connect to the external Festo OPC UA server and return the requested node.
    """
    c = Client(endpoint)
    c.connect()
    n = c.get_node(nodeid)
    return c, n

# CREATE LOCAL OPC UA VARIABLES
def create_equipment_node(parent_node, equipment_name, namespace_idx):
    """
    Create an OPC UA object node and add all EMS variables that will be
    updated
    from Arduino sensor data and Festo comparison data.
    """
    equipment_node = parent_node.add_object(namespace_idx, equipment_name)
    tags = {}

    tag_list = [
        "AtmosphericTemperature",
        "AtmosphericPressure",
        "AirFlow",
        "AtmosphericHumidity",
        "DiffPressure",
        "PneumaticPressure_bar",
        "Current",
        "Voltage",
        "Watts",
        "Festo_ActivePower_L1_W"
    ]

    # Create each variable as a float node
    for tag_name in tag_list:
        tags[tag_name] = equipment_node.add_variable(
            namespace_idx,
            tag_name,
            ua.Variant(0.0, ua.VariantType.Float)
        )
        tags[tag_name].set_writable()

    return tags

```

```

# MAIN PROGRAM
if __name__ == "__main__":

    # START LOCAL OPC UA SERVER
    # where EMS sensor values will be published
    server = Server()
    server.set_endpoint("opc.tcp://localhost:4840/freeopcua/server/")
    server.set_server_name("EMS Server")

    # namespace for this EMS project
    idx = server.register_namespace("EMS_Server")

    # root object and attaching EMS variables
    objects = server.get_objects_node()
    EMS_data_node = objects.add_object(idx, "EMS Data")
    tags = create_equipment_node(EMS_data_node, "EMS", idx)

    # START CSV LOGGING
    # Open CSV file and write column headings
    csv_file = open(csv_filename, "w", newline="", encoding="utf-8")
    csv_writer = csv.writer(csv_file)

    csv_writer.writerow([
        "timestamp_iso",
        "AtmosphericTemperature_C",
        "AtmosphericPressure_Pa",
        "AirVelocity_mps",
        "Voltage_V",
        "Current_mA",
        "Power_W",
        "Humidity_pct",
        "DiffPressure_Pa",
        "PneumaticPressure_bar",
        "Festo_ActivePower_L1_W"
    ])
    csv_file.flush()

    # CONNECT TO FESTO OPC UA
    # compare EMS data with built-in Festo power data
    festo_client = None
    festo_node = None
    festo_value_node = None
    festo_power_w = float("nan")

    try:
        festo_client, festo_node = connect_festo(FESTO_ENDPOINT,
        FESTO_NODE_ACTIVE_POWER_L1)

```

```

        print("Connected to Festo OPC UA")
    except Exception as e:
        print("WARNING: Could not connect to Festo OPC UA:", e)

    # Start local OPC UA server
    server.start()
    print("OPC UA Server running")
    print(f"Connect UAExpert → opc.tcp://localhost:4840/freeopcua/server/")
    print(f"Logging CSV → {csv_filename}")

    rows_written = 0
    last_festo_poll = 0.0

    try:
        while True:

            # POLL FESTO POWER NODE
            # Read Festo AC power at a slower interval than Arduino serial
data
            now = time.monotonic()
            if now - last_festo_poll >= FESTO_POLL_SECONDS:
                last_festo_poll = now

                if festo_client and festo_node:
                    try:
                        # numeric child node already found read directly from
it
                        if festo_value_node:
                            festo_power_w =
float(festo_value_node.get_value())
                        else:
                            raw_val = festo_node.get_value()

                            # If returned value is an ExtensionObject search
for numeric child
                            if isinstance(raw_val, ua.ExtensionObject):
                                festo_value_node =
resolve_numeric_child_node(festo_node)
                                if festo_value_node:
                                    festo_power_w =
float(festo_value_node.get_value())

                            # If returned value is already numeric use it
directly
                            elif isinstance(raw_val, (int, float)):
                                festo_power_w = float(raw_val)

                    # Update local OPC UA tag with Festo power

```

```

        tags["Festo_ActivePower_L1_W"].set_value(festo_power_w
    )

    except Exception:
        # If read fails clear connection objects so script can
continue safely

        festo_client = None
        festo_node = None
        festo_value_node = None

    # READ ARDUINO SERIAL DATA
    # Read one CSV line from Arduino
    line = arduino.readline().decode(errors="ignore").strip()
    parsed = parse_9_float_csv(line)

    # Skip invalid lines and continue
    if parsed is None:
        time.sleep(0.02)
        continue

    # Unpack parsed sensor values
    (
        tempC,
        pressurePa,
        airVel_ms,
        voltageV,
        current_mA,
        power_W,
        humidity_pct,
        diffPressurePa,
        pneuPressure_bar
    ) = parsed

    # UPDATE LOCAL OPC UA TAGS
    # Publish live EMS values to local OPC UA server
    tags["AtmosphericTemperature"].set_value(tempC)
    tags["AtmosphericPressure"].set_value(pressurePa)
    tags["AirFlow"].set_value(airVel_ms)
    tags["AtmosphericHumidity"].set_value(humidity_pct)
    tags["DiffPressure"].set_value(diffPressurePa)
    tags["PneumaticPressure_bar"].set_value(pneuPressure_bar)

    tags["Voltage"].set_value(voltageV)
    tags["Current"].set_value(current_mA)
    tags["Watts"].set_value(power_W)

    # LOG DATA TO CSV
    # Save each measurement row with timestamp

```

```

        ts = datetime.now().isoformat(timespec="milliseconds")
        csv_writer.writerow([
            ts,
            tempC,
            pressurePa,
            airVel_ms,
            voltageV,
            current_mA,
            power_W,
            humidity_pct,
            diffPressurePa,
            pneuPressure_bar,
            festo_power_w
        ])

        # Flush file every few rows to reduce risk of losing data
        rows_written += 1
        if rows_written % LOG_EVERY_N_ROWS == 0:
            csv_file.flush()

        # PRINT LIVE DEBUG OUTPUT
        # Show current sensor values in terminal for monitoring during
tests
        print(
            f"T={tempC:.2f}°C | P_atm={pressurePa:.0f} Pa | "
RH={humidity_pct:.1f}% | "
            f"AirVel={airVel_ms:.3f} m/s | dP={diffPressurePa:.2f} Pa | "
            f"Pneu={pneuPressure_bar:.3f} bar | "
            f"V={voltageV:.2f} V | I={current_mA:.2f} mA | "
Pdc={power_W:.3f} W | "
            f"FestoAC={festo_power_w:.2f} W"
        )

        # Main loop delay
        time.sleep(0.05)

    finally:
        # CLEAN SHUTDOWN
        # Stop server close serial port flush/close CSV and disconnect Festo
client
        try:
            server.stop()
        except Exception:
            pass

        try:
            arduino.close()
        except Exception:

```

```

        pass

    try:
        csv_file.flush()
        csv_file.close()
    except Exception:
        pass

    try:
        if festo_client:
            festo_client.disconnect()
    except Exception:
        pass

    print("Server stopped.")

```

8.4 Data filtering code

```

import pandas as pd
import numpy as np
import os

# Path to the raw CSV file containing the test data
INPUT_CSV = r"C:\Users\sheuz\OneDrive\Desktop\OPC Data\Idle test.csv"

# Window size
# Used to remove spikes and outliers from the current signal
WINDOW_MED = 5

# Window size moving average filter
# Used to smooth the signal after median filtering
WINDOW_MA = 5

# Lower and upper clipping limits
# Current values below 0 A are treated as invalid and removed
# CLIP_MAX no upper limit is applied
CLIP_MIN = 0.0
CLIP_MAX = None

# Read the CSV file into a DataFrame
df = pd.read_csv(INPUT_CSV)

# Check that the current column exists in the dataset
if "Current_mA" not in df.columns:
    raise ValueError("Column 'Current_mA' not found in CSV")

```

```

# Copy the raw current values so the original data is preserved
I_raw = df["Current_mA"].copy()

# Clamp the current values to remove negative readings
# If CLIP_MAX is not provided only the lower bound is applied
if CLIP_MAX is None:
    I_clamped = np.clip(I_raw, CLIP_MIN, np.inf)
else:
    I_clamped = np.clip(I_raw, CLIP_MIN, CLIP_MAX)

# Apply a centred median filter to remove spikes and isolated noise
I_med = I_clamped.rolling(window=WINDOW_MED, center=True,
min_periods=1).median()

# Apply a centred moving average filter to smooth the signal
I_filt = I_med.rolling(window=WINDOW_MA, center=True, min_periods=1).mean()

# Store the filtered current signal in a new column
df["Current_mA_filtered"] = I_filt

# If a voltage column exists calculate filtered power using:
# Power = Voltage x Current
# Current is converted from mA to A
if "Voltage_V" in df.columns:
    df["Power_W_filtered"] = df["Voltage_V"] * df["Current_mA_filtered"] /
1000.0

# Create the output filename by adding "_filtered"
base, ext = os.path.splitext(INPUT_CSV)
OUTPUT_CSV = base + "_filtered" + ext

# Save the processed DataFrame to a new CSV file
df.to_csv(OUTPUT_CSV, index=False)

# Confirm where the filtered file was saved
print("Saved filtered file to:", OUTPUT_CSV)

```