POLITECNICO DI TORINO

Master of Science in Mechatronic Engineering

December 2025

Design and Validation of a Modular Motion Control System for a Spectral Characterization Bench

Candidate

Thesis Supervisor
prof. CHIABERGE MARCELLO

Industrial Tutor
BERAUD PASCAL

MINACCIATTI ENZO

Abstract

This thesis, conducted in collaboration with Sodern–ArianeGroup, addresses the modernization of a spectral characterization bench dedicated to the qualification of space-grade optical filters—a key instrument ensuring spectral accuracy and stability in aerospace applications. The legacy motion control subsystem, based on proprietary hardware and undocumented software, had become a critical bottleneck in terms of maintainability, precision, and scalability.

To overcome these limitations, a modular industrial motion control architecture was designed and validated, centered on a Zaber X-MCC controller. The work followed a complete engineering workflow, including system analysis, design and wiring of a two-axis prototype, hardware–software co-integration, and the development of modular control interfaces using LabWindows/CVI. The proposed framework enables precise stepper motor actuation, incremental encoder feedback, and digital triggering of electromechanical devices, all within cleanroom-compatible constraints.

Experimental validation on Sodern's optical metrology infrastructure confirmed both compliance with industrial requirements (electrical, mechanical, and environmental) and a significant improvement in operational performance: a representative spectral scan was completed in thirty minutes instead of five hours. Beyond its functional validation, this work contributes a scalable and reproducible methodology for industrial motion control applied to optical metrology benches, providing a foundation for future multi-axis implementations and broader test system standardization across space instrumentation platforms.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Mr. Pascal Béraud, my industrial supervisor at Sodern-ArianeGroup, for his invaluable guidance, technical insight, and continuous support throughout this work. His expertise in mechatronic systems and his availability at every stage of the project were essential to its successful completion.

I also wish to extend my sincere thanks to Mr. Franck Pécal and to all the members of the Special Products Department for their warm welcome, constructive discussions, and daily collaboration. Their professionalism and openness created an ideal environment for both learning and innovation.

I am equally grateful to my academic supervisors and faculty members from ENSIL-ENSCI and Politecnico di Torino, whose advice, encouragement, and scientific guidance made this joint academic–industrial work possible.

Finally, I would like to thank my colleagues and fellow interns at Sodern for their camaraderie, enthusiasm, and support, which made this experience both technically enriching and personally memorable.

Table of contents

Introduction

Context and Motivation

Sodern is a French high-tech company specializing in optronics, neutron instrumentation, and space optics. Founded in 1962 as an offshoot of Philips' electronics and applied physics laboratories, the company was originally created to provide neutron sources for France's strategic defence program.

Over the decades, Sodern has diversified its activities to become a global reference in several strategic domains. In the late 1960s, the company expanded into ground-based optical systems, and later into space applications, notably with the development of star trackers (Fig. 1), for which Sodern is now a world leader.

Figure 1: Daylight optical sight – star trackers

Today, Sodern is a subsidiary of ArianeGroup (90% ownership) and partially held by the Commissariat à l'Energie Atomique et aux Energies Alterntives (CEA, French Alternative Energies and Atomic Energy Commission) (10%). This affiliation reinforces its position in the international aerospace market and its ability to innovate in high-tech sectors. The company employs around 450 people (Fig. 2), over 60% of whom are engineers, and is structured into six main departments.

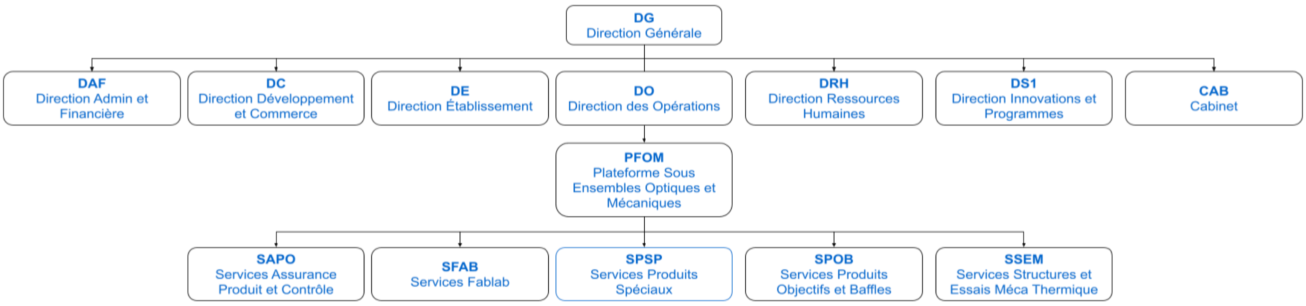


Figure 2: Organisation chart

Introduction

Sodern plays a key role in space exploration, Earth observation, national defense, and scientific instrumentation, collaborating closely with major institutions such as:

- The European Space Agency (ESA) – for the supply of optical instruments on multiple space missions
- The French space agency CNES – for R&D partnerships
- Defense industries – for the development of advanced military-grade optronic systems

Sodern currently equips approximately one-third of all international satellites and spacecraft, including telecommunication satellites, Earth climate monitoring programs, scientific payloads, and deep-space missions. It is also the world leader in neutron generators, and a pioneer in neutron interrogation instruments. One example is the FastGrade™ probe (Fig. 3), which allows subsurface exploration and material analysis using neutron technology.

Figure 3: FastGrade™

In the field of optical instrumentation, Sodern develops and manufactures focal plane arrays, high-performance space cameras, and contributes to ambitious scientific missions such as:

- The PHARAO atomic clock (onboard the ACES mission)
- The SEIS seismometer deployed on Mars by NASA's InSight mission

Among its key industrial test tools is a custom spectral characterization test bench (Fig. 4), designed to qualify multispectral optical filters (Fig. 5) used in space instruments. These filters are essential for precise spectral band selection, especially for Earth observation satellites and scientific payloads requiring strict optical calibration and reliability.

Over the years, the test bench has remained a central element of Sodern's qualification process. However, its original architecture (both hardware and software) began to show signs of aging. The need for improved maintainability, industrial scalability, and long-term sustainability became increasingly evident.

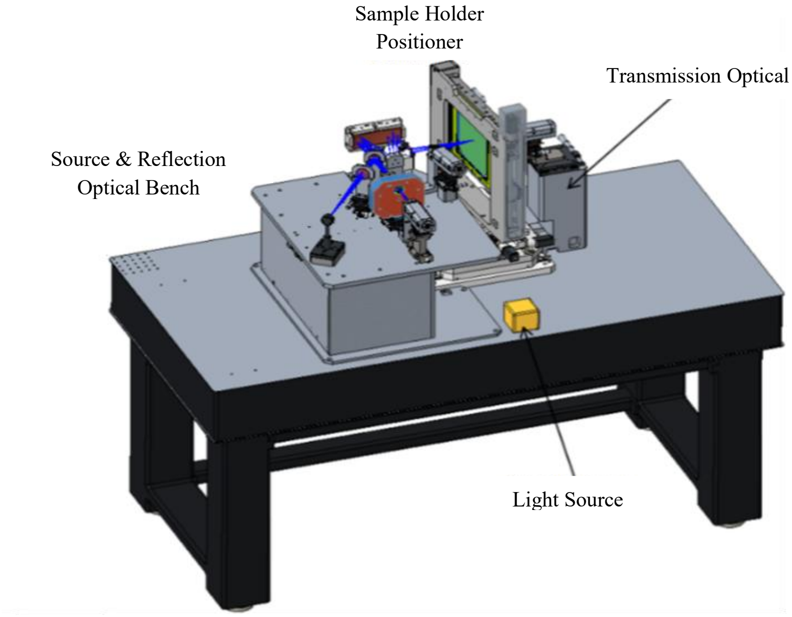


Figure 4: Spectral Characterization Test Bench

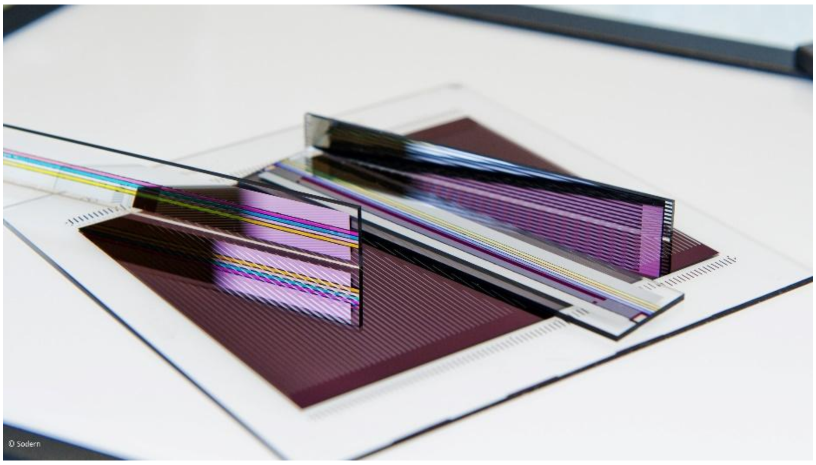


Figure 5: Multispectral optical filters qualified by the bench

In this context, Sodern launched a modernization initiative aimed at updating and improving this critical infrastructure. The project falls within a broader strategy of industrial optimization to meet growing demands in both Space and Defense markets.

It was within this technical and strategic framework that I carried out an internship, in the Special Products Department of the Opto-Mechanical Systems Division. My work contributed to the early stages of this modernization effort, focusing on the motion control aspect of the test bench and supporting its future industrial upgrade.

Problem Statement and Objectives

The modernization of the spectral characterization bench raised several technical challenges. Beyond replacing obsolete components, the project required the design of a new motion control architecture capable of meeting strict constraints of precision and reliability in a cleanroom environment.

The legacy setup, based on a proprietary ISP control rack and a LabVIEW interface, had become increasingly unreliable and difficult to maintain. To address these limitations, Sodern initiated a transition toward a new industrial architecture based on standardized hardware and structured software tools. The work was organized around the following technical goals:

- Hardware architecture validation
    - Evaluate the compatibility of a new industrial motion controller with the existing electromechanical components.
    - Design and assemble a functional prototype.
    - Adapt the electrical interfaces.
    - Configure each axis.
- Software migration preparation
    - Develop standalone LabWindows/CVI applications in C/C++, capable of sending commands to the motion controller.
    - Implement executables that can initialize the bench and launch simple automated measurement sequences.
- Integration and documentation
    - Validate the prototype under realistic operating conditions to confirm the robustness of the new hardware/software stack
    - Confirm that replacing the control rack successfully addressed all obsolescence concerns and resulted in noticeable performance improvements.
    - Deliver a complete technical specification for the final industrial five-axis drawer to be outsourced for production.

Methodology

The work followed a structured engineering methodology inspired by the V-cycle approach:

- Phase 1: System Analysis and Reverse Engineering

Study of technical documentation, mechanical diagrams, electrical schematics, and the LabVIEW-based software previously used to control the bench. Reverse engineering of the current system's logic and signals.

- Phase 2: Prototype Design and Development

Electrical wiring of a two-axis prototype. Integration of the new motion controller. Development of a modular LabWindows/CVI codebase.

- Phase 3: Testing and Validation

Incremental testing of communication, movement commands, and feedback mechanisms. Evaluation of positioning accuracy and consistency. Logging and debugging.

- Phase 4: Documentation and Specification

Drafting of technical documentation for rack design and LabWindows/CVI code. Preparation of subcontracting deliverables and project handover to the Sodern engineering team.

Overview of the Developed System

The final prototype developed during the thesis consisted of:

- A partial-scale motion control rack capable of managing two motorized axes (out of five required in the final system).
- LabWindows/CVI interfaces offering automated motion control and serial communication diagnostics.
- A framework for trajectory execution, position feedback reading, and system calibration.

The two-axis prototype and software tools provide a basis for testing key functionalities such as motor command, sensor feedback, actuator triggering, and communication with the host PC. This configuration aims to enable broader validation of performance, safety, and system integration readiness for future deployment on the full five-axis configuration.

1. State of the Art

1.1. Spectral Characterization of Optical Filters for Space Applications

Optical filters are fundamental components in space instrumentation, enabling the selective transmission or rejection of specific wavelength bands. This spectral isolation is essential in applications such as Earth observation, planetary science, and Defense imaging, where the quality of data depends heavily on precise spectral discrimination.

In recent years, the demand for high-performance optical filtering in space payloads has increased with the proliferation of hyperspectral and multispectral missions, leading to new requirements on both spectral accuracy and spatial uniformity. These advances have also shifted the focus from purely optical design to system-level characterization, where mechanical positioning, optical alignment, and environmental control all contribute to the uncertainty budget.

Modern space payloads frequently use multispectral or hyperspectral imaging systems, in which filters define individual spectral channels. These filters directly affect the system's spectral resolution, radiometric accuracy, and overall performance [1].

High-performance optical filters are typically fabricated using thin-film interference technologies, where alternating layers of dielectric materials are deposited onto substrates to create constructive or destructive interference effects. Techniques such as ion beam sputtering or plasma-assisted evaporation are used to achieve high optical quality and environmental stability [2]. However, despite technological advancements, spatial non-uniformities in coating thickness can still occur, particularly over large apertures [3].

The performance of such filters is commonly assessed in terms of spectral accuracy ($\Delta\lambda \leq 1$ nm) and spatial uniformity (<1%), which depend not only on coating quality but also on the precision of the characterization apparatus itself. Therefore, the metrological integrity of the test bench—its ability to position samples repeatably, control illumination geometry, and maintain environmental cleanliness—becomes a determining factor in overall mission readiness.

To address these issues, manufacturers produce large filter wafers which are then diced into smaller elements, commonly called "matchstick filters." Each filter must be individually characterized, as even minor coating variations can alter its optical response [2].

Spectral characterization, in this context, refers to the precise measurement of transmission, reflection, and sometimes polarization as a function of wavelength, and in some cases, spatial position across the filter's surface. For space-grade components, this characterization must be both spectrally accurate and spatially resolved [3].

Standard commercial systems such as scanning spectrophotometers and monochromators are often used for this purpose. However, due to constraints in geometry, throughput, or custom filter formats, these are not always suitable for industrial qualification workflows. Previous studies have proposed custom characterization setups, but few describe the engineering architecture that ensures both optical accuracy and operational robustness in an industrial context. This gap motivates the present work, which aims to bridge metrological precision and industrial maintainability within the same system. As a result, several research centers and companies, including Sodern, have developed in-house test benches optimized for the specific constraints of space filter characterization [3].

1.2. Previous Work and Bench Design

The spectral characterization bench used at Sodern was co-developed with ISP Systems as part of an internal R&D program aimed at testing optical interference filters for space applications. Its architecture combines high optical precision, mechanical flexibility, and environmental cleanliness, allowing for repeatable characterization of thin-film filters under realistic operating conditions.

The Sodern–ISP Systems bench represents an evolution of earlier laboratory concepts toward a semi-industrial platform, capable of meeting both research-grade precision and production-level repeatability. While the optical chain and environmental control had reached maturity, the motion control and software architecture remained a limiting factor for reliability and scalability. This observation guided the present modernization effort.

1.2.1. Optical System Overview

The core of the bench is an advanced optical chain (Fig. 6) designed to analyse the spectral behaviour of filters in both transmission and reflection. A high-intensity light source coupled to an optical fibre delivers broadband illumination, which is collimated through a diaphragm (pinhole) to produce a spot of calibrated size on the sample surface.

After interacting with the optical component under test, the light is either:

- Transmitted, and collected via a second fibre aligned to a spectrometer, or
- Reflected, and redirected through a second optical path including a mirror relay (Offner relay).

A beam splitter is used to divert a small portion of the flux to a photodiode, allowing for real-time monitoring of source stability during acquisition. This is essential to normalize spectral data, especially in long-duration scans. [4]

Such sensor feedback, typical of precision test benches, provides redundancy and enables flux normalization and drift compensation, key aspects for traceable optical metrology.

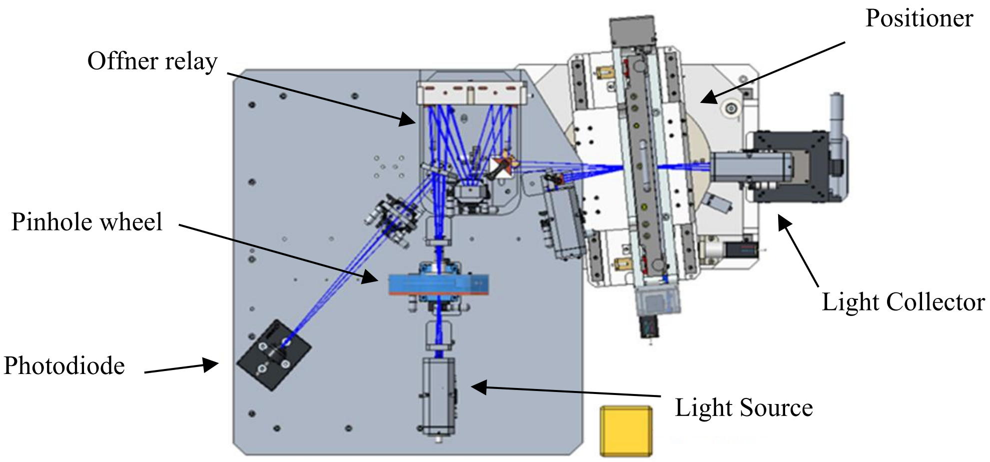


Figure 6: Spectral Bench seen from above

A motorized pinhole wheel (Fig. 7) allows dynamic adjustment of the beam aperture by selecting among six precision pinholes (50 µm to 1000 µm diameter), enabling different spot sizes depending on the filter geometry or required spatial resolution.

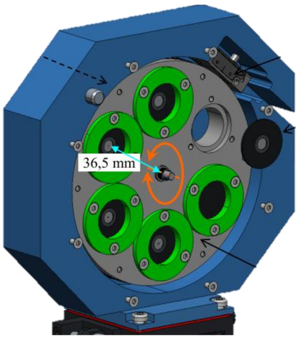


Figure 7: Pinhole Wheel

A high-resolution inspection camera, mounted on a dedicated arm, is used to observe micron-scale defects on the filter surface. Illumination is provided by a barlight LED, electronically controlled through the bench control rack.

1.2.2. Sample Positioning System

The bench features a four-axis sample positioner (Fig. 8), allowing precise motion of the filter component to scan areas of interest:

- X-axis translation (transverse motion),
- Y-axis translation (along the optical axis),
- Z-axis translation (vertical motion, to scan height).
- Theta-axis rotation (vertical rotation aligned with the optical surface),

Each motion axis is driven by the same type of geared motor. Filters are mounted inside a cassette with rolling guides, ensuring repeatable and safe insertion.

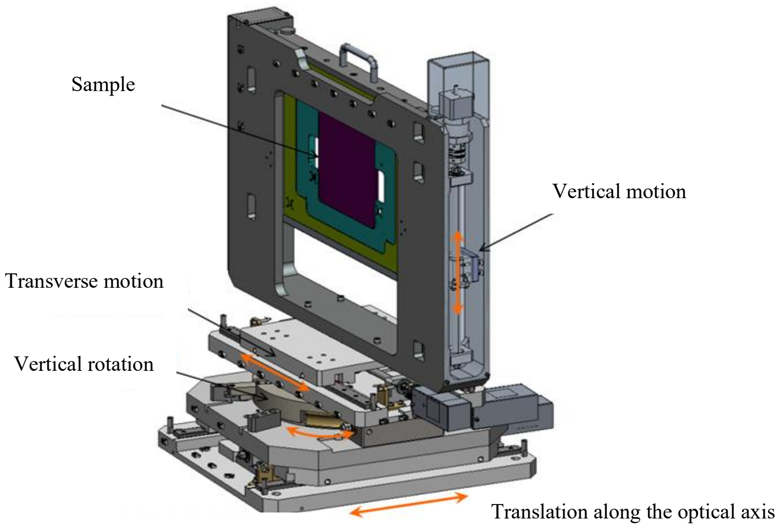


Figure 8: Positioner

However, this mechanical versatility also introduces stringent requirements on axis synchronization, homing repeatability, and thermal stability—aspects directly influenced by the underlying motion control system.

1.2.3. Signal Acquisition and Data Flow

The light collected from the sample is fed into an Optical Spectral Analyzer (Fig. 9), located in a 19-inch rack close to the optical table. An optical switch allows automatic toggling between reflection and transmission channels.

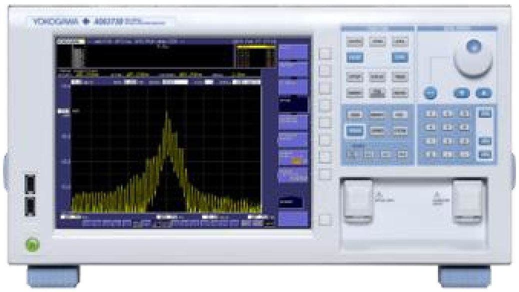


Figure 9: OSA

All measurement parameters, axis trajectories, and acquisition sequences are managed through a custom LabVIEW-based Human-Machine Interface (Fig. 10). This software allows the operator to define position coordinates and scanning paths, synchronize motion with spectral acquisition, monitor flux in real time via the photodiode, and visualize camera images.

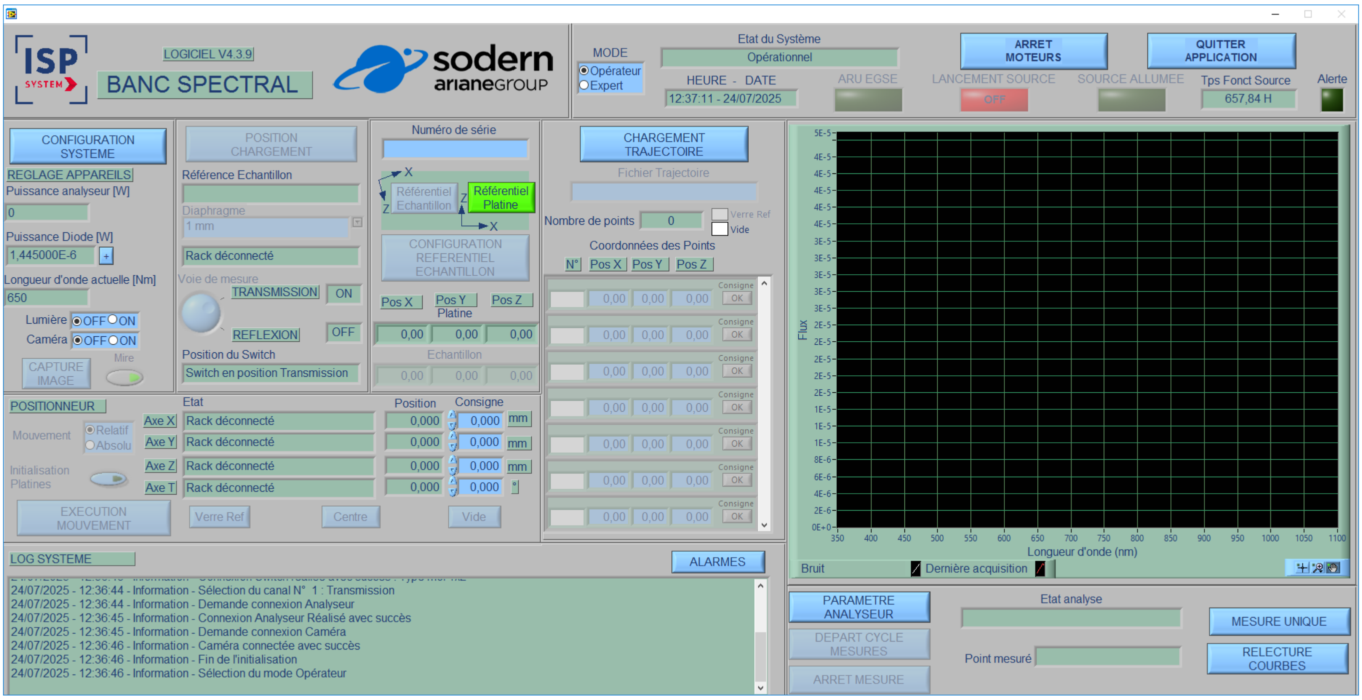


Figure 10: HMI of the Spectral Bench

1.2.4. Cleanroom Compliance

All mechanical and optical components of the bench were selected or adapted for ISO6 cleanroom compatibility, using stainless steel or anodized aluminum parts, Braycote 601EF lubricant or dry ceramic guides for motion stages, and capotted motors with shielded wiring to minimize particle generation.

1.3. Motion Control and Software Limitations

The original motion control system relied on a custom-built rack. Motion was generated through trapezoidal profiles with limited fine-tuning options.

The legacy system, designed around a proprietary rack and LabVIEW-based software, presented a typical example of a "black-box" architecture: limited observability, non-standard interfaces, and low reproducibility of motion trajectories. Such constraints are particularly problematic in a cleanroom environment, where repeatability and autonomy are essential.

Over time, the system exhibited increasing reliability issues:

- Position loss (stall) over long trajectories.
- Frequent homing failures, with axes not consistently returning to reference positions.
- Intermittent blocking on the vertical axis, likely due to inadequate stall detection and power regulation [5].

In addition to hardware limitations, the LabVIEW-based software environment posed several challenges:

- The CAN-based communication protocol between the rack and PC was difficult to monitor and debug.
- Persistent software bugs, especially in trajectory loading and error handling, impacted daily operations and required repeated manual interventions.
- No unified logging system was available to track motor behaviour or encoder values in real time, complicating diagnostics.

Overall, the aging hardware and the complexity of maintaining undocumented code created significant technical debt, hindering both daily usage and future scalability. From a research perspective, these limitations highlight a gap between laboratory-grade motion control—optimized for flexibility—and industrial motion architectures, which prioritize determinism, traceability, and long-term maintainability. Bridging this gap forms one of the central engineering challenges addressed in this thesis.

1.4. Justification for a New Architecture

Faced with the growing unreliability of the motion control system and the difficulty with the legacy software stack, a complete architectural redesign was required. This redesign was not only an engineering update but a methodological opportunity to define a modular, controller-agnostic motion control framework, applicable to various optical test benches. The new solution had to improve reliability during long and repetitive operations, simplify the software stack with a structured and maintainable programming language and ensure long-term maintainability through documented interfaces. To achieve these goals, Sodern initiated a transition toward an industrial architecture based on a new motion controller with encoder feedback and TCP/IP ASCII communication, a migration from LabVIEW to LabWindows/CVI for more maintainable and traceable software, and a modular rack-based design with clear electrical interfaces and reusable code structure. From an academic standpoint, the modernization aims to demonstrate that industrial architectures can meet metrological requirements traditionally reserved for laboratory systems, by combining standardized communication protocols (ASCII over TCP/IP), modular code structure, and feedback-based calibration.

This strategic shift illustrates Sodern's commitment to test bench standardization and software robustness for space-grade optical component testing. This project thus contributes to the broader research question of how to design maintainable yet metrologically traceable motion systems for space optics applications.

1.5. Related Technological References

The modernization of the motion control system for the spectral characterization bench at Sodern was guided by established practices in the aerospace instrumentation sector. In this domain, precise and repeatable motion is typically achieved through closed-loop systems based on stepper or servo motors, monitored by incremental or absolute encoders, and controlled via industrial-grade motion controllers implementing trajectory planning and feedback regulation.

Comparative studies across commercial systems show that the key differentiators between architectures lie not in raw motion speed, but in protocol transparency, ease of calibration, and logging capability—elements directly impacting diagnosability and certification readiness.

Standard architectures in optical test environments share several recurring characteristics. Hardware modularity is ensured through rack-based motion systems with Sub-D connectors, which simplify cable standardization and allow rapid component replacement, while relays, sensors, and actuators are typically driven by open-collector outputs with solid-state signal adaptation [6]. Communication between controllers and supervisory PCs is usually established over TCP/IP, RS232, or RS485 links. Some manufacturers, such as Zaber or Thorlabs, favour ASCII command sets [6][7], while others like Galil or Moog Animatics rely on register-based APIs, offering both integration flexibility and scripting compatibility. On the software side, graphical tools such as LabVIEW are often replaced in professional optical benches by more maintainable compiled environments like LabWindows/CVI or ANSI C/C++ [8].

The approach adopted at Sodern aligns with this trend by emphasizing traceable configurability (via human-readable parameters), standardized TCP/IP communication, and robust feedback integration. This positions the developed architecture as both an industrial solution and a reproducible experimental platform for future research in motion control applied to optical metrology.

1.6. Research Gap and Objectives

While the optical and mechanical aspects of spectral characterization benches have been thoroughly investigated in the literature, their motion control subsystems are rarely described from a system-engineering standpoint. Most publications focus on optical performance—spectral resolution, uniformity, or environmental stability—without addressing how the underlying motion architecture affects these results in terms of positioning accuracy, repeatability, and long-term reliability. Consequently, there is a lack of documented methodologies for designing motion systems that are simultaneously metrologically traceable, industrially maintainable, and scalable to multi-axis configurations.

This thesis aims to bridge that gap by developing and validating a modular motion control architecture dedicated to spectral characterization of space optics.

2. Project Specifications and Execution

This chapter presents the detailed engineering work carried out to design, implement, and validate the new motion control system developed for Sodern's spectral characterization bench. The project aimed to replace an obsolete control subsystem with a modular, maintainable, and industrial-grade architecture (see the section Problem Statement and Objectives, p. 4). The structure reflects the complete engineering workflow — from requirements definition to validation — illustrating the interdisciplinary nature of the work between electronics, software, and optical metrology.

2.1. System Requirements and Integration Constraints

This section defines the functional, electrical, and environmental requirements that guided the conception of the new motion control system. These constraints — arising from the bench's optical precision needs and its existing hardware interfaces— directly influenced the architectural choices and technical decisions made throughout the project.

2.1.1. Functional Requirements

The spectral characterization bench requires precise and repeatable movements of optical components along its five motorized axes of the positioner. The new motion control system therefore had to support the existing bipolar stepper motors, integrate incremental encoders for position feedback, and correctly process digital limit switch signals for end-of-travel detection. It also needed to ensure TCP/IP communication with the bench PC, remain fully compatible with the LabWindows/CVI software environment, and provide digital outputs to drive electromechanical actuators.

2.1.2. Electrical Interface Compatibility

To ensure seamless integration with Sodern's existing infrastructure, the new motion control solution had to interface with several predefined hardware elements, already installed and operational on the bench.

2.1.2.1. Stepper Motors

- Motors: 4 × MS14HS1P4070 stepper (Fig. 11) (200 steps/rev, 0.7 A/phase, 0.1 Nm) + 1 × AM1524 micro-stepper (24 steps/rev; for the pinhole wheel).
- Gearboxes: 4 × PM32 0020 (19.2:1) + 1 × 546:1 planetary (AM1524).
- Connection: SUB-D15 connectors with non-standard pinout (phases $A/\bar{A}$, $B/\bar{B}$, …).

Figure 11: Stepper Motors

2.1.2.2. Encoders

- Encoders: Renishaw QUANTiC incremental (Fig. 12), RKLC40-S scale, 50 nm resolution.
- Signals: Differential RS-422 (A/B/Z) at +5 V with shielded twisted-pair wiring.

Figure 12: Renishaw QUANTiC incremental encoder

2.1.2.3. End-of-Travel Switches

- Type: Magnetic REED switches (Fig. 13)
- Logic level: Referenced to +24 V
- Function: Used for limit detection and homing operations, defines the minimum/maximum allowed position.
- The simultaneous activation of both sensors is mechanically impossible, ensuring robust detection of travel limits during normal operation.

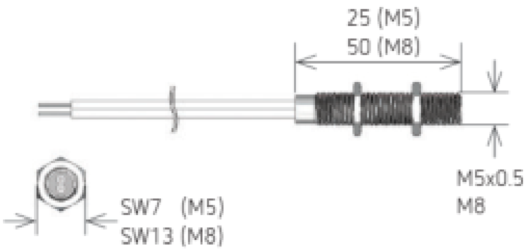


Figure 13: End-of-Travel sensors

2.1.2.4. Electromechanical Actuators

- Usage: Trigger on/off components
- Output control requirement: The system must support external actuation through reliable digital output.
- Load characteristics:
  - Nominal voltage: 24 VDC
  - Current per channel: up to 300 mA

These interface requirements were considered non-negotiable, and had to be supported as-is by the new architecture.

2.1.3. Communication and Software Integration

Another essential design requirement concerned software compatibility with LabWindows/CVI, the development environment routinely employed at Sodern for instrument control and data acquisition. This constraint imposed several functional expectations on the new motion controller. It had to ensure deterministic Ethernet (TCP/IP) communication, support modular software architecture to allow future expansion toward a full five-axis configuration or integration of additional instruments such as spectrometers and cameras, and rely on human-readable command protocols to facilitate debugging and interoperability. Furthermore, the system needed to accommodate both low-level command-line testing and high-level graphical user interface (GUI) operation. Consequently, the selected controller was required to provide straightforward socket-based communication capabilities as well as comprehensive documentation and software libraries to ensure reliable integration within Sodern's existing software ecosystem.

2.2. Hardware Architecture and Electrical Interfaces

The implementation of a new motion control system for Sodern's spectral characterization bench necessitated a comprehensive redesign of the hardware interfaces linking the controller to the bench's motors, actuators, sensors, and encoders. This section presents a detailed description of the control rack architecture and the signal adaptation strategies employed to guarantee full compatibility with the pre-existing electromechanical infrastructure.

2.2.1. Motion Controller Selection

The selection of the motion controller was carried out under the guidance of the project supervisor, who recommended the Zaber X-MCC (Fig. 14) based on its previous successful use in other Sodern systems and its proven reliability in similar applications. Already familiar to the development team, this controller represented a low-risk and well-established solution.

Technically, the X-MCC2 model met the requirements of the spectral characterization bench by providing native support for the bipolar stepper motors already in use, integrated handling of quadrature encoders (RS-422), and Ethernet communication through a simple ASCII protocol, which facilitated integration with LabWindows/CVI. It also offered open-collector outputs for triggering external devices and a compact industrial-grade design suitable for cleanroom installation. The controller supports up to two independent axes with closed-loop feedback, and configuration as well as diagnostic operations are performed via Zaber Launcher, enabling setup, encoder calibration, and firmware updates.

Figure 14: Zaber X-MCC2

2.2.2. Rack Architecture

For prototyping purposes, the motion control equipment was arranged as a temporary standalone setup (Fig. 15) next to the spectral bench bay. This open configuration allowed easy access to all components during integration and testing, and served as a functional equivalent of the future rack-based system.

The assembly included:

- A Zaber X-MCC2 motion controller (2-axis version),
- A 24 VDC industrial power supply (PS12S-24V30, 3 A max),
- Two individual solid-state relays (Crydom DRA1-MP) fixed to the controller.

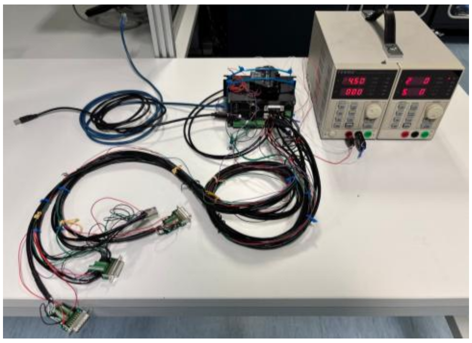


Figure 15: Prototype layout

The assembly replicated the expected electrical and functional behavior of the future industrial drawer. This flexible arrangement made it possible to test motor control, sensor interfaces, and actuator triggering under realistic conditions.

2.2.3. Stepper Motor Wiring and Compatibility

Electrical interfacing (Fig. 16) between the spectral bench and the motion controller required a custom cable assembly to connect the existing SUB-D15 motor connectors coming from the bench to the Zaber SUB-D26 motor inputs.

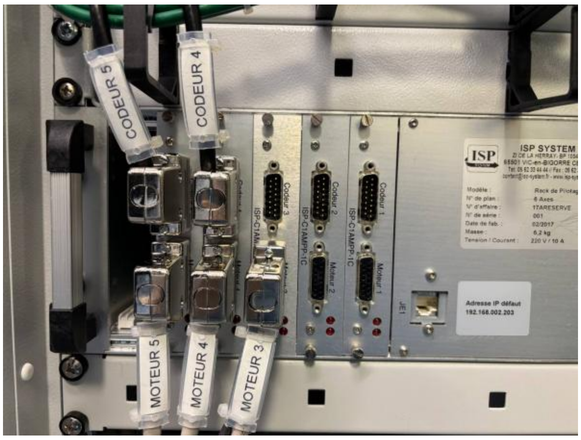


Figure 16: Legacy Interface between the bench and the obsolete ISP rack

The four motor phase wires (A, $\overline{A}$, B, $\overline{B}$) were routed according to the pinout defined in the Zaber documentation and matched with Sodern's SUB-D15 assignment. *[Complete pinout mapping: see **Appendice A.1**]*

The wiring was manually adapted by connecting the phase signals to the appropriate differential motor outputs on the SUB-D26 connector. Brake and Over-temperature pins on the controller side were left unconnected, since they are not used in the current bench setup.

On the bench side, screw terminal blocks (Fig. 17) were used to allow modular maintenance and future rework. The controller-side wiring was manually soldered, with each joint insulated using heat-shrink tubing for mechanical and electrical protection.

The controller supports bipolar stepper motors with up to 3 A per phase and 48 V supply voltage, which meets all requirements for the motors present on the spectral characterization bench.

2.2.4. Encoder Signal Adaptation

The incremental encoders used on each axis deliver differential RS-422 signals across six lines (A, $\overline{A}$, B, $\overline{B}$, Z, $\overline{Z}$), requiring a noise-immune, high-integrity transmission path to the motion controller. All differential signal pairs were directly routed to the corresponding encoder input pins on the SUB-D26 connector of the Zaber controller, based on the datasheet specifications. *[Complete pinout mapping: see **Appendice A.1**]*

The encoders were powered via the +5 V supply line provided by the controller and shielded twisted-pair cables (from an Ethernet cable) were used to carry the signals, with the shielding grounded on the controller side to prevent ground loops and minimize EMI (electromagnetic interference).

The encoder wiring followed a custom pinout conversion between Sodern's SUB-D15 connector (Fig. 17) and the controller's SUB-D26 input. The Z reference signal used for homing is routed as part of the encoder cable, and its processing is entirely managed by the Zaber firmware configuration (does not need to be routed to the Zaber Home Limit sensor pin).

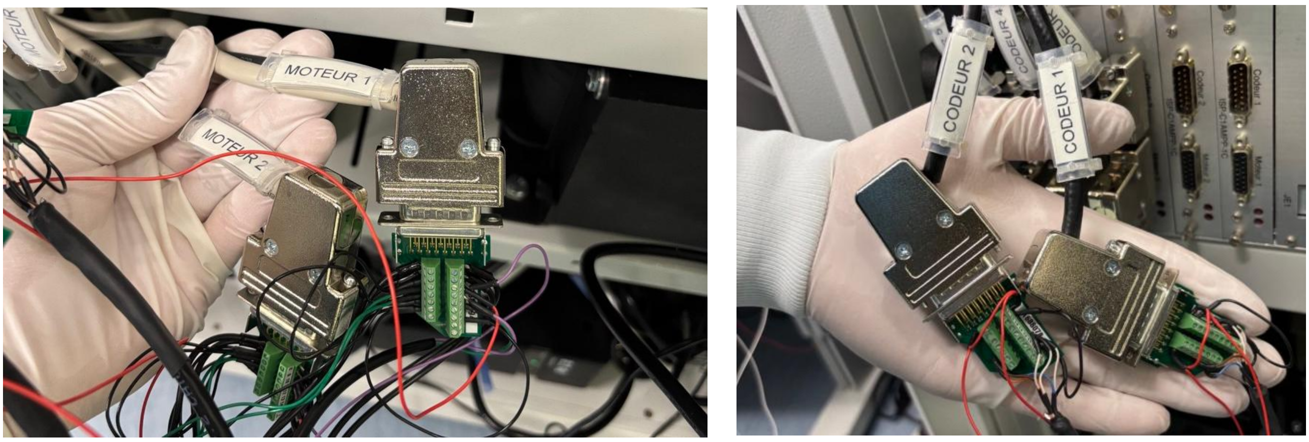


Figure 17: Bench motor and encoder connectors

2.2.5. End-of-Travel Sensor Logic Conversion

The magnetic REED end-of-travel sensors from the legacy ISP rack were reused with minor wiring adaptations. Originally referenced to +24 V as normally-closed switches, they were reconfigured for the Zaber X-MCC2, which requires ground-referenced, open-collector inputs. *[Complete pinout mapping: see* ***Appendice A.1****]*

In the prototype rack:

- The common line was tied to ground.
- Each sensor signal was connected directly to the Zaber limit inputs.
- Homing was managed via encoder Z reference, as no dedicated home sensor exists.

2.2.6. Electromechanical Actuation via Relays

The digital outputs of the Zaber X-MCC2 controller are open-collector and can only sink a few milliamps of current (insufficient to directly drive Sodern's 24 V electromechanical coils). To adapt these outputs, individual solid-state relays Sensata Crydom DRA1-MP (Fig. 18) were used.

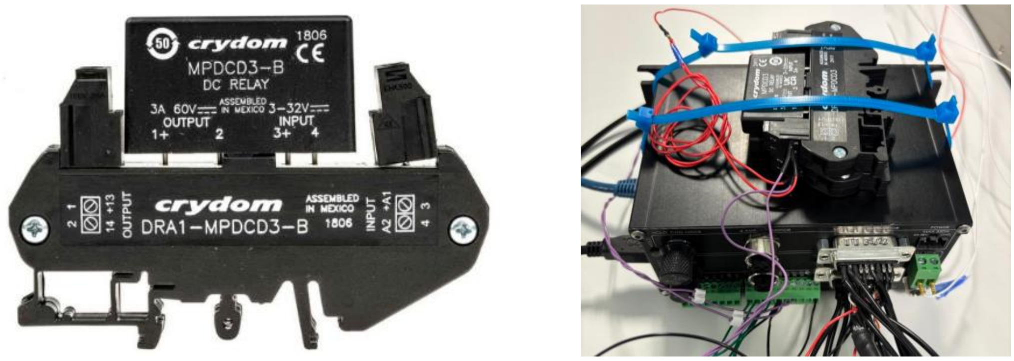


Figure 18: Solid-state relays connected to Zaber output

In order to validate the use of the solid-state relay as an interface between the Zaber controller digital outputs and the bench's actuators, a series of laboratory measurements were conducted. A resistive load of 1.8 kΩ was used to emulate the bench coil (Fig. 19).

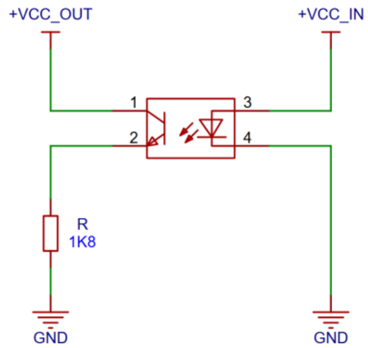


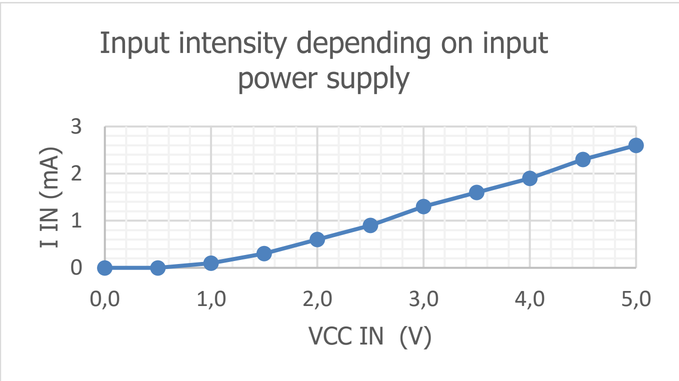
Figure 19: Solid-state relay measurement setup schematic

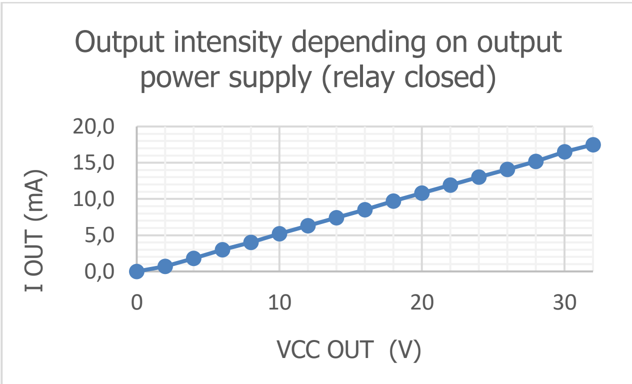
2.2.6.1. Input Current vs. Input Voltage

The input voltage (VCC_IN) was swept from 0 to 5 V while measuring the input current. The resulting curve was linear, with a threshold intercept at approximately 1.2 V and a slope of about 0.6 mA/V. This behaviour corresponds to the expected forward voltage of the infrared LED used inside the relay and the internal 1.5 kΩ series resistor specified in the datasheet (see the Input characteristic below).

2.2.6.2. Output Current vs. Output Voltage

With the input held constant at 5 V, the output voltage (VCC_OUT) was varied between 0 and 32 V. The measured output current increased linearly, again with a coefficient of approximately 0.6 mA/V. No clear lower limit in the output current was observed, confirming that the relay can operate reliably even at very low drive levels.





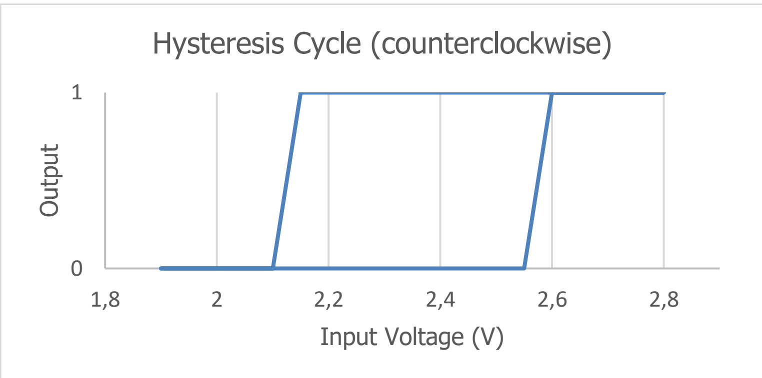
Input and Output characteristics

2.2.6.3. Leakage Current Test

When VCC_IN was set to 0 V and VCC_OUT fixed at 24 V (relay in the OFF state), the leakage current was measured: I_{OUT} (leak) = 0,32 mA. The observed value was within acceptable limits and did not risk false triggering of the coils.

2.2.6.4. Hysteresis Behaviour

A hysteresis cycle was characterized by varying VCC_IN around the switching point. The transition from logic 0 to logic 1 occurred at approximately 2.5 V, while the reverse transition occurred at around 2.1 V. This small hysteresis window ensures stable switching and avoids chatter around the threshold.



The tests confirmed that the input side of the relay behaves as expected. On the output side, the device shows no practical lower limit on the driven current, which is advantageous for compatibility with various actuators. Based on these results, this solid-state relay reference is validated as suitable for use in the spectral bench control rack.

2.2.7. Cable Assembly and Electrical Validation

To interface the motion controller with the existing components of the spectral characterization bench, all wiring was custom-fabricated using industrial-grade materials and manual assembly techniques.

A dedicated one-axis cable (Fig. 20) was assembled for each channel, connecting the bench's SUB-D15 connectors to the controller's SUB-D26 inputs and digital output headers.

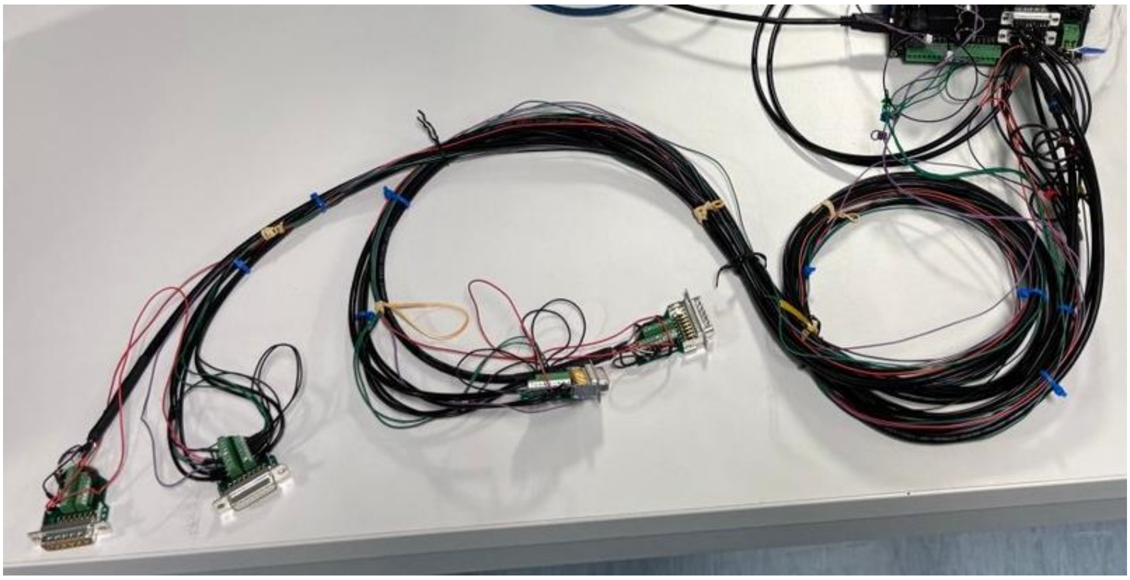


Figure 20: Complete two-axis cable with SUB-D15 and SUB-D26 connectors

Finally, every wire was checked for correct routing and end-to-end connection using a multimeter. The complete electrical diagram of one-axis cable is available in **Appendice A.2** and can be reused for all future channels in the five-axis implementation.

2.3. Prototype Implementation and Testing

In order to assess the feasibility of the new motion control system prior to its full-scale deployment on the five-axis spectral characterization bench, a reduced-scale prototype incorporating the hardware architecture described in the previous section was developed. This section presents the practical implementation of the prototype and the subsequent evaluation of its performance.

2.3.1. Objectives of the Prototype

The prototype was implemented as a two-axis control system representative of the final bench. Its purpose was to verify electrical compatibility between the Zaber controller and existing components (motors, encoders, limit sensors), validate TCP/IP communication using ASCII commands, and test actuator triggering via relays. It also allowed debugging of motion logic (homing, limit detection, encoder feedback) and evaluation of positioning accuracy and repeatability. The setup (Fig. 21) combined a Zaber X-MCC2 controller powered at 24 VDC, the bench (stepper motors with encoders), custom interface cables for axis wiring, solid-state relays to drive 24 VDC coils, and a PC workstation connected over Ethernet.

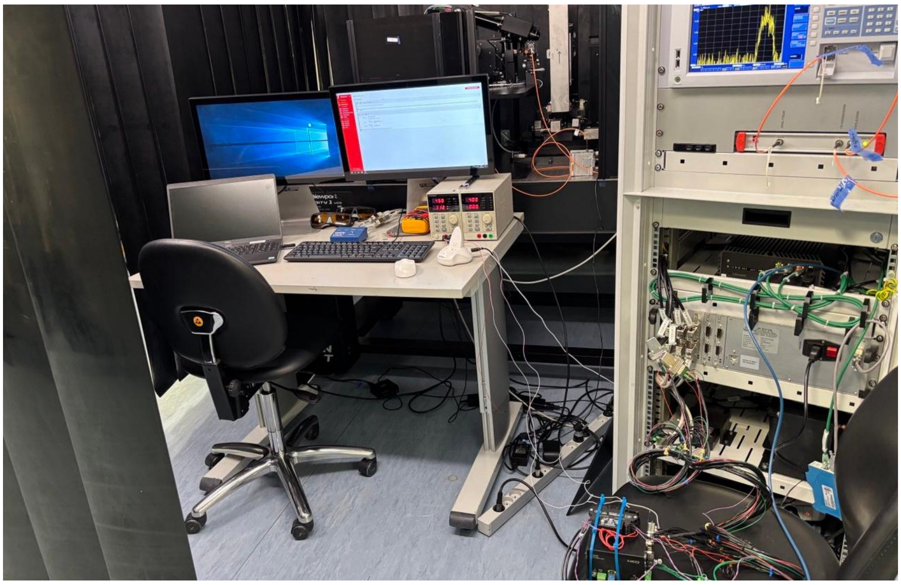


Figure 21: Prototype setup on the bench

2.3.2. Controller Configuration and Parameter Tuning

Before any movement or feedback operation could be executed, the Zaber X-MCC motion controller had to be configured using Zaber Launcher, the manufacturer's official software suite. This tool provides a graphical interface to define low-level hardware parameters (motor, encoder, driver), as well as motion constraints, safety limits, and closed-loop behaviour. This section details the tuning of the controller parameters as applied to the two-axis prototype.

2.3.2.1. Advanced Hardware Setup and Motor Profile Identification

Before configuring the motors in Zaber Launcher (Fig. 22), the non-Zaber stepper motors of the bench had to be characterized using the Advanced Hardware Setup utility. This tool identifies third-party motors and encoders, generating a Device ID later linked to the axis configuration. It ensures correct motion profiles, current regulation, and encoder feedback. For each motor, type, drive mode, encoder interface, mechanical reduction, electrical parameters, and resolution were specified based on datasheets and scale specifications.

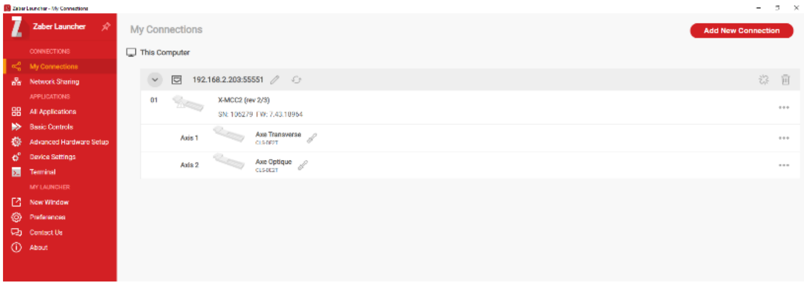


Figure 22: Zaber Launcher Software

Once confirmed, this configuration generated a profile with a unique identifier that was later loaded into the Zaber X-MCC2 controller firmware for each axis.

2.3.2.2. Microstepping Theory and Control Principles

Precise motion control in stepper motors relies on the ability to divide each mechanical step into smaller angular increments. This technique, known as microstepping, is implemented in nearly all modern industrial motion controllers, including the Zaber X-MCC2 used in this project. It enables smooth, low-vibration motion and sub-micrometer positioning accuracy, essential for optical metrology applications such as the spectral characterization bench.

In a conventional stepper motor, two stator windings (A and B) are energized in discrete sequences to move the rotor by fixed angular steps—typically 1.8° per full step (200 steps per revolution). In microstepping mode, instead of switching the windings fully on or off, the controller continuously varies the current in each phase according to a sinusoidal profile (Fig. 23):

$$I_A = I_{max} \times \sin(\theta)$$

$$I_B = I_{max} \times \cos(\theta)$$

where I_A and I_B are the instantaneous currents in the two motor phases, I_{max} is the rated phase current, and θ is the electrical phase angle corresponding to the desired microstep position.

By incrementing θ in small steps (for example, 1/8, 1/16, or 1/64 of a full step), the magnetic field vector rotates smoothly within the stator, causing the rotor to follow intermediate equilibrium positions between the standard step detents.

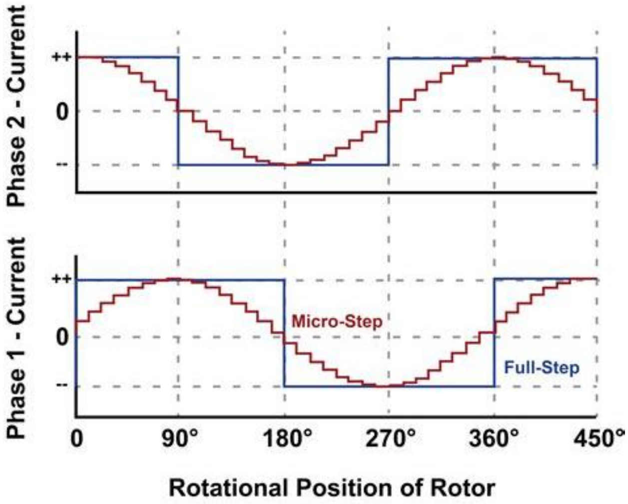


Figure 23: Current Wave Form of Micro-Step vs Full-Step

Industrial motion controllers such as the Zaber X-MCC generate these phase currents using PWM (Pulse Width Modulation) and a current-mode feedback loop. An internal digital controller measures the actual winding current and adjusts the PWM duty cycle to follow the ideal sinusoidal reference. This process is handled by a PID-based inner control loop, ensuring accurate current tracking even under varying load conditions. In practice, this means that for each microstep command, the controller regulates both phase currents to their exact target values in real time, effectively synthesizing a continuous magnetic field rotation.

The main benefit of microstepping lies in the smoother torque and reduced vibration compared to full-step or half-step operation. Each microstep creates a nearly continuous torque curve, avoiding the resonance phenomena commonly observed in discrete stepping.

Key advantages include high positional resolution (the angular increment per microstep is reduced by a factor of the microstepping ratio, e.g., 1/64 = 0.028° per step for a 1.8° motor), reduced mechanical stress (smoother torque transitions lower wear and improve reliability), lower noise and vibration (critical in optical benches), and improved linearity (the combination of sinusoidal current control and encoder feedback allows repeatable sub-micrometer motion).

While microstepping improves smoothness and resolution, it does not increase absolute torque proportionally. The available torque per microstep is approximately proportional to the sine of the microstep angle, meaning that the holding torque per increment decreases with finer subdivision. For this reason, high microstepping ratios (e.g., 1/128 or 1/256) may degrade positional stiffness if not paired with closed-loop correction.

In practice, the effective motion accuracy also depends on the linearity of current regulation in the driver, the mechanical stiffness and backlash of the transmission, and the precise calibration of the microstep-to-distance conversion factor.

The Zaber X-MCC combines open-loop microstepping with an outer closed-loop position control based on encoder feedback. In this architecture, the inner current loop maintains accurate sinusoidal excitation of motor phases, while the outer loop monitors encoder feedback and compensates for position errors such as stall, lost steps, or mechanical backlash. This dual-loop structure effectively transforms the stepper motor into a hybrid servo system, achieving the smoothness of microstepping with the reliability of servo control. Such control is especially relevant for optical metrology benches, where sub-micrometer repeatability is required over long scanning sequences.

In summary, microstepping allows stepper motors to achieve quasi-continuous motion by modulating phase currents sinusoidally. When implemented in an industrial controller such as the Zaber X-MCC, this technique provides smooth, precise, and quiet actuation, well suited for metrology and optical test systems. The accuracy of the displacement, however, still depends on the correct calibration of the microstep conversion factor and the feedback quality of the encoder loop. Together, these elements form the foundation of the motion control approach adopted for the spectral characterization bench.

2.3.2.3. Microstep Resolution and Motion Conversion Factor

The Zaber X-MCC2 motion controller natively operates in microsteps, which means that all motion commands are expressed in units of microsteps, not millimeters or radians. To interpret and control physical motion with high accuracy, it is therefore essential to calculate the conversion factor between microsteps and actual displacement.

For the three main linear axes of the spectral bench (X, Y and Z), the mechanical transmission follows the same structure (see 2.1.2.1. Stepper Motors):

- Motor: 200 steps/rev
- Gearbox: 19.2:1 planetary reduction
- Lead screw pitch: 2 mm

Each full motor revolution corresponds to:

$$Displacement\ per\ revolution\ = \frac{screw\ pitch}{reduction} = \frac{2\ [mm]}{19{,}2} = 0{,}10417\ mm$$

Thus, one full step (1/200 revolution) gives:

$$Displacement\ per\ motor\ step\ = \frac{Displacement\ per\ revolution}{step\ per\ revolution} = \frac{0{,}10417\ [mm]}{200} = 0{,}52\ \mu m/step$$

With a microstepping resolution of 1/64, the controller operates at:

$$Microstep\ resolution = \frac{0{,}52\ [\mu m]}{64} = \mathbf{8{,}125\ nm/\mu step}$$

This theoretical value becomes the fundamental unit of the three linear axes motion in all software control layers.

Using the same type of calculation for the two other rotations:

- The Theta rotation stage for optical sample alignment uses a 130-tooth gear system.
  - Motor: 200 steps/rev
  - Gearbox: 19:1
  - Gear ratio: 130:1

  ➔ $Angular\ resolution = \frac{2\pi}{200 \times 19 \times 130 \times 64} = 0.198\ \mu rad/\mu step$

- The pinhole wheel is driven by a stepper gearmotor
  - Motor: 24 steps/rev
  - Gear ratio: 546:1

➔ $Angular\ resolution = \frac{2\pi}{24 \times 546 \times 64} = 7.49\ \mu rad/\mu step$

2.3.2.4. Configuration of Main Parameters

The motor configuration within Zaber Launcher was organized according to its primary settings tabs. In the Driver section, the run current was set to 0.66 A (peak) in accordance with the motor specifications, while the hold current was reduced to 0.2 A (peak) to minimize heating. Microstepping was maintained at 64 µsteps per step, providing a balance between positioning precision and motion smoothness. Encoder readings were utilized directly in their raw form, without filtering or offset correction.

In the Closed-Loop tab, stall detection was maintained at its default value of 200 ms with a tolerance of ±128 µsteps (approximately 1 µm), and the feedback ratio was adjusted to 5/52 to ensure accurate correspondence between commanded steps and actual travel, with 50 encoder counts per 520 motor steps. The Current Controller operated under the factory PID settings, which provided stable performance without requiring manual tuning.

Motion parameters were configured with both acceleration and deceleration set to 1 mm/s^2, and the maximum speed limited to 1 mm/s. Software-defined limits were established in microsteps according to the stroke of each axis, preventing mechanical overtravel and ensuring safe operation within the physical constraints of the bench.

2.3.2.5. Limit and Direction Configuration

The configuration of motion directions and end-of-travel limits was a fundamental step to ensure consistent and reliable operation of the spectral characterization bench. Motor orientation was determined by phase wiring, driver direction, and encoder counting, and was adjusted for each axis to maintain a consistent reference frame, preserve legacy interface conventions, and guarantee accurate homing. In cases where mechanical constraints fixed the motor cabling, directional adjustments were performed in software by modifying the sign of the commands in microsteps.

Each axis was equipped with two REED limit sensors, designated as Close and Away, which were rewired to be compatible with the Zaber controller inputs. The sensor logic was configured as Active High, so that inactive sensors read 0 V, indicating “not reached,” and active sensors read 2.5 V, indicating “reached.” The mechanical design ensured that both sensors could not be triggered simultaneously, providing robust and reliable detection of travel limits.

2.3.2.6. Homing Strategy and Initialization Constraints

The axis initialization step, or homing, is essential in any precision positioning system as it defines a stable reference for all subsequent movements. On Sodern’s spectral characterization bench, the absence of a dedicated Home sensor, the proximity of some Close Limit sensors to the homing zone, and the presence of multiple Z index marks every 50mm on the Renishaw encoder scale required the development of a robust strategy.

The full homing sequence consists of disabling the control loop, moving the axis slightly in the positive direction until the limit sensor is reached, then launching the homing command while returning in the negative direction and finally activate the control loop. The first Z mark encountered is registered as the absolute zero and stored by the controller. This controlled procedure avoids incorrect index detection and provides a consistent initialization, even in the presence of multiple Z marks. Tests confirmed that this strategy delivers repeatable, reliable homing without requiring additional mechanical sensors, laying the foundation for future automation of the bench's startup process.

2.3.3. Functional Testing and Validation

The prototype drawer was subjected to a complete validation campaign to evaluate the behaviour of the motion control system in real conditions. The evaluation focused on key aspects, including motor control accuracy and repeatability, the integrity of feedback signals, and the logic of actuator triggering. Each subsystem was evaluated independently to ensure the system would be ready for full-scale deployment on the spectral characterization bench.

2.3.3.1. Motor Control

All tests described in this section were performed on an optical filter (wafer) mounted on the positioner. The component was observable through the bench's inspection camera, and its dimensions were obtained from a technical document provided by an expert (Fig. 24). This setup ensured that all motion measurements could be cross-checked against fixed visual references on the component surface.

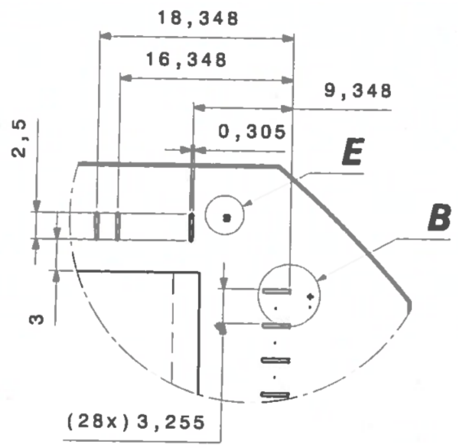


Figure 24: Wafer dimensions extract (dimension in mm)

Test 1 – Encoder Resolution Validation

Objective: Confirm that the encoder resolution corresponds to the theoretical specification of 50 nm per count, using a physically calibrated travel distance.

Protocol:

- The mobile axis was aligned manually with two distinct optical markers spaced 32.550 mm apart
- The encoder position was read at the initial and final points using the encoder counts on the Zaber controller.
- The number of counts was computed as the absolute difference between start and end.

- The effective size per count was then computed as:

$$Size\ per\ count\ [nm] = \frac{Measured\ distance\ [mm] \times 10^6}{Encoder\ counts}$$

Results:

Measure	Start (count)	End (count)	Δ(count)	Distance (mm)	Size per count (nm)
1	-964356	-1615223	650867	32.543	49.99
2	-964387	-1615241	650854	32.5427	49.99
3	-1615241	-964368	650873	32.5436	49.99
4	-1615221	-964378	650843	32.5421	49.99

Analysis: The encoder consistently reports approximately 650,850 counts over a 32,543 mm motion, yielding an average resolution of:

$$\frac{32{,}543 \times 10^6}{650850} = 49{,}99\ nm/count$$

This validates the encoder's specification, with error below ±0.01%. It also confirms the correctness of the scaling factor applied in the controller configuration.

Test 2 – Absolute Positioning Accuracy in Open Loop

Objective: Evaluate the open-loop precision by comparing a commanded movement to the position actually reached, as measured by the encoder.

Protocol:

- The theoretical movement required to cover 32.550 mm was calculated from the known theoretical microstep resolution 8,125 nm/µstep (see 2.3.2.2. Microstep Resolution and Motion Conversion Factor):

$$\frac{32{,}550}{8{,}125 \times 10^{-6}} = 4006154\ \mu step$$

- Command sent: */1 1 move rel 4006154*
- Positions were measured with the encoder counts displayed on Zaber Launcher.

Results:

Measure	Reached (count)	Target (count)	Error (count)	Error (nm)
1	-1295688	-1294752	-936	-46.8
2	-1295692	-1294750	-942	-47.1
3	-1295690	-1294750	-940	-47.0

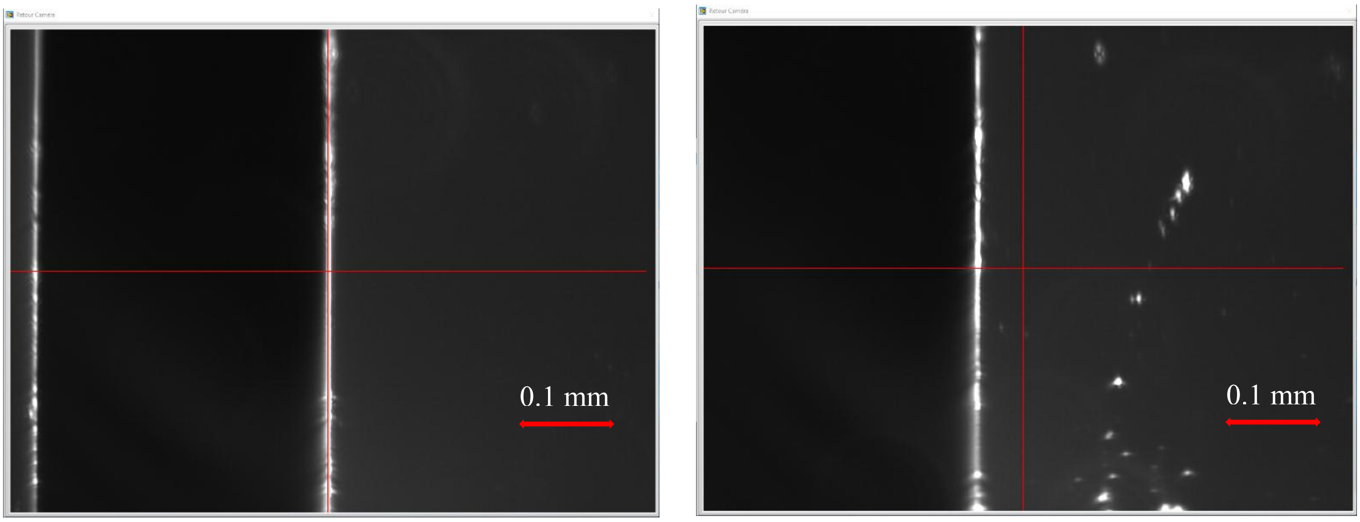


Figure 25: Deviation in Open Loop (starting position on the left)

Analysis: The observed deviation (Fig.25) of approximately 940 encoder counts (47.0 µm) does not reflect a mechanical inaccuracy of the motion system, but rather a mismatch between the theoretical displacement per microstep used for the command calculation (8.125 nm/µstep) and the actual value calibrated from the encoder. In this test, the motion command was derived purely from the nominal gearbox ratio, screw pitch, and microstepping resolution, without prior experimental adjustment of the conversion factor used to calculate the command in µstep.

Such a discrepancy is expected when theoretical parameters are used without empirical calibration, and can be eliminated by updating the steps-per-unit configuration to match the measured encoder scaling. Once calibrated, open-loop motion would converge much more closely to the target position.

Calibration: Following this analysis, the system was recalibrated by updating the steps-per-unit factor used to calculate the proper command value in microstep. The effective displacement per microstep was measured experimentally from encoder data over a long travel, then averaged to minimize the influence of local noise. The calibrated conversion factor was stored: 8,113nm/µstep, ensuring that commanded displacements in microsteps now match the actual encoder-measured travel. After this calibration, the residual error was reduced to below a few micrometers, confirming the accuracy of the adjustment.

Test 3 – Closed-Loop Positioning Accuracy

Objective: Assess positioning accuracy when encoder feedback is enabled.

Protocol:

- The same target motion as Test 2 was executed, but with closed-loop feedback control.
- The position reached was verified visually against the reference marker using camera inspection.

Results:

Starting (count)	Reached (count)	Visual Target	Visual Error
-964324	-1615353	Marker aligned	< Visual precision

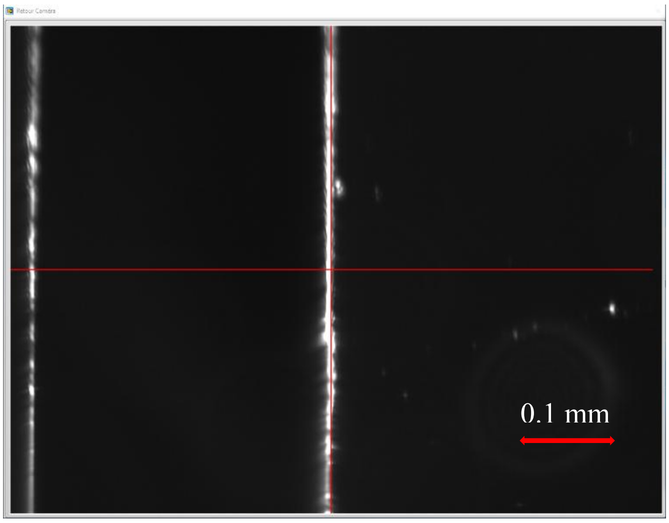


Figure 26: Alignment in Closed-Loop Positioning (starting and reached positions are exactly the same alignment)

Analysis: The alignment (Fig.26) was visually perfect using the external marker. Any remaining deviation was below the optical resolution of the measurement setup (≈50 counts or 2.5 µm), confirming the effectiveness of encoder-based feedback. The controller correctly compensates for the systematic error observed in open-loop mode.

Test 4 – Repeatability Without Closed-Loop Correction

Objective: Validate the ability of the system to return to the same position after multiple cycles, without encoder-based closed-loop correction.

Protocol:

- The axis was moved +32.550 mm and then back to the start, multiple times.
- The encoder position was recorded before and after each cycle.

Results:

Cycle	Start (count)	Return (count)	Error (count)	Error (nm)
1	-643970	-643963	7	350
2	-643979	-643966	13	650

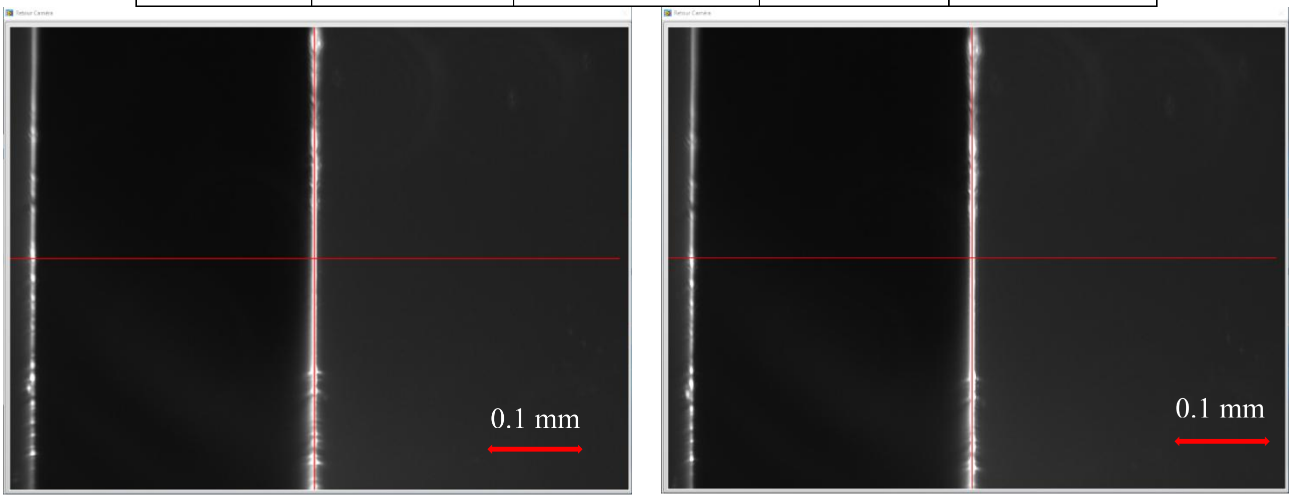


Figure 27: Repeatability Without Closed-Loop Correction (starting position on the left)

Analysis: The repeatability (Fig. 27) was within 15 encoder counts (750 nm), which is excellent for a system operating without closed-loop correction. This reflects a high mechanical stability of the transmission chain and confirms that thermal drift and backlash are negligible over short cycles.

Test 5 – Homing Repeatability Using Encoder Z Reference

Objective: Evaluate the repeatability of the homing sequence when using the encoder's Z reference mark.

Protocol:

- The axis was homed repeatedly from different initial positions.
- After each homing, the encoder count at the home position was recorded.

Results:

Home	Reached (count)
1	-622070
2	-622069

Home	Reached (count)
3	-622069
4	-622070

Analysis: The home position was reached with a repeatability better than ±1 count (±50 nm). This confirms that the configured homing strategy is robust, consistently locking onto the same physical Z mark, and is suitable for high-precision alignment operations.

2.3.3.2. Sensor Feedback

The end-of-travel sensors (magnetic REED switches) were tested on the prototype to verify their electrical behaviour (Fig. 26) and compatibility with the Zaber X-MCC controller's digital inputs. Measurements were carried out using an oscilloscope connected directly to the relevant input pins on the controller (e.g., Close Limit pin).

Test 1 – Transition through the sensor

Objective: Verify the voltage transitions (Fig. 28) when the axis passes over the magnetic trigger zone.

Protocol: The tested axis was moved from the middle of its travel (sensor inactive) towards the Close Limit position until the sensor was activated, then moved back to the inactive zone. The oscilloscope probe was connected to the Close Limit Sensor pin, referenced to ground.

Results:

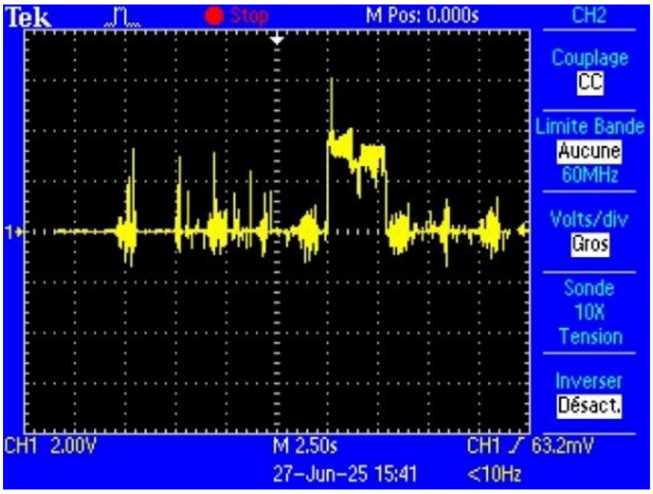


Figure 28: Close Limit Sensor signal during activation and release

In the inactive state, the sensor holds the input at 0 V; when activated by the magnet, the contact opens and the voltage rises to ~2.5 V (Zaber's internal pull-up). Leaving the magnetic zone brings the input back to 0 V. The signal shows high-frequency noise, but this does not affect reliable logic detection.

Test 2 – Signal Behavior in Prolonged Active State

Objective: Observe and characterize the voltage signal delivered by the end-of-travel sensors (Fig.29) when remaining in the active (triggered) state for several seconds.

Protocol: Two oscilloscope channels were connected simultaneously to the Close Limit inputs of both prototype axes. Each axis was positioned so that its respective REED sensor remained continuously triggered, and the voltage at the controller input was recorded over time.

Results:S

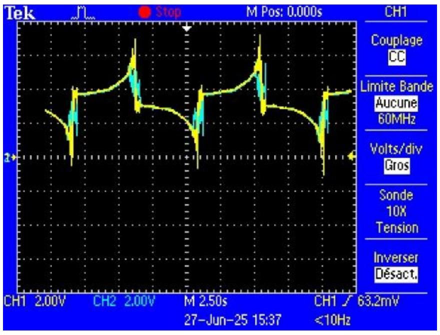


Figure 29 : Periodic voltage fluctuation observed on two active Close Limit inputs

Both channels exhibited an identical periodic variation with a total cycle time of approximately 10 seconds.

- First 5 seconds: slow exponential rise from ~2.4 V to ~2.8 V.
- Last 5 seconds: symmetric exponential decay back to ~2.4 V.
- The two channels remained perfectly in phase, indicating a common origin rather than sensor-specific behavior.

Analysis: This periodic drift is not characteristic of a passive magnetic REED switch, which should provide a stable logical level when closed or open. The synchronized behavior across both inputs strongly suggests that the fluctuation originates from the internal circuitry of the Zaber controller, rather than from the sensors or the cabling. According to the controller datasheet, "limit sensor inputs are pulled up to an internal supply rail and are designed to be pulled low by an open collector". This internal pull-up network likely includes RC elements or multiplexed sensing circuitry, resulting in the observed slow charge/discharge cycles.

Importantly, these voltage oscillations remain above the HIGH detection threshold of the controller's digital input stage and therefore do not impact detection reliability during axis motion.

2.3.3.3. Actuator Triggering

The objective of this test was to verify the correct operation of the electromechanical actuators (coils) when controlled via the Zaber X-MCC2 digital outputs and the solid-state relay interface (see 2.2.6. Electromechanical Actuation via Relays).

Electrical measurements were performed under two conditions:

Command (Dig.out)	Voltage at Zaber output	Voltage at the coil
0 (OFF)	4.44 V	0.48 V
1 (ON)	0.19 V	23.75 V

Analysis:

When the digital output is set to OFF, the Zaber output remained high (~4.44 V), preventing current to flow in the relay input diode, and so preventing the swich activation. In this state, the coil line measured approximately 0.48 V, confirming that no load is applied.

When the digital output is set to ON (triggered via Zaber Launcher), the Zaber output is pulled low (~0.19 V), causing the relay to close and apply voltage to the TOR coil. The coil voltage goes to 23.75V during activation, consistent with the load characteristics and the internal voltage drop of the relay.

2.3.4. Observations and Adjustments

The experimental validation campaign provided critical insights into the configuration of the Zaber X-MCC controller and the mechanical transmission of the spectral characterization bench prototype. The Renishaw encoders consistently achieved their nominal resolution of 50 nm per count, confirming both intrinsic sensor accuracy and the integrity of the signal path. Initial observations in open-loop mode revealed a systematic deviation over long commanded travels, which had been addressed previously through the calibration procedure described in Section 2.3.3.1. This calibration ensured that commanded microstep displacements closely matched the actual encoder-measured travel, significantly reducing residual positioning errors.

Repeatability tests further demonstrated the high mechanical stability of the transmission chain. Even without encoder-based correction, return positioning errors remained below 750 nm, indicating minimal backlash and negligible thermal drift over short motion cycles.

Closed-loop testing, configured through an appropriate feedback ratio, effectively compensated for the remaining drift. This operational mode requires precise initialization of the encoder's index (Z) channel, emphasizing the necessity of a reliable homing sequence for high-accuracy performance. These findings underscore the importance of the homing strategy described in Section 2.3.2.5, particularly the consistent definition of motion directions and the targeting of a unique Z reference mark.

2.4. Software Development and Debug Tools

To ensure reliable communication with the motion controller, facilitate the integration of the control system into the spectral characterization bench, and enable preliminary automated testing representative of the final operational procedure, a set of dedicated software tools was developed using LabWindows/CVI 2019.

This software suite includes two distinct Human–Machine Interfaces (HMIs) and an automated measurement program running in the background, designed to perform measurement sequences autonomously. The development process followed a layered approach, allowing for progressive validation of each subsystem and ensuring full consistency with the modular control architecture defined by Sodern.

The following subsections describe the design principles, implementation strategy, and validation of these software components.

2.4.1. Controller Debug HMI

The first Human Machine Interface (Fig. 30) developed during the internship was designed as a lightweight debugging tool to test communication with the Zaber X-MCC controller independently of other bench subsystems. Its primary purpose was to validate TCP/IP connectivity, ASCII protocol compliance, and command-response integrity before integrating the controller into the bench.

The interface included a command input field, allowing the operator to manually enter any valid ASCII command defined in the Zaber protocol. An output log console displayed the raw responses returned by the controller, including both success messages and error codes. Visual indicators were also provided and show the current connection status.

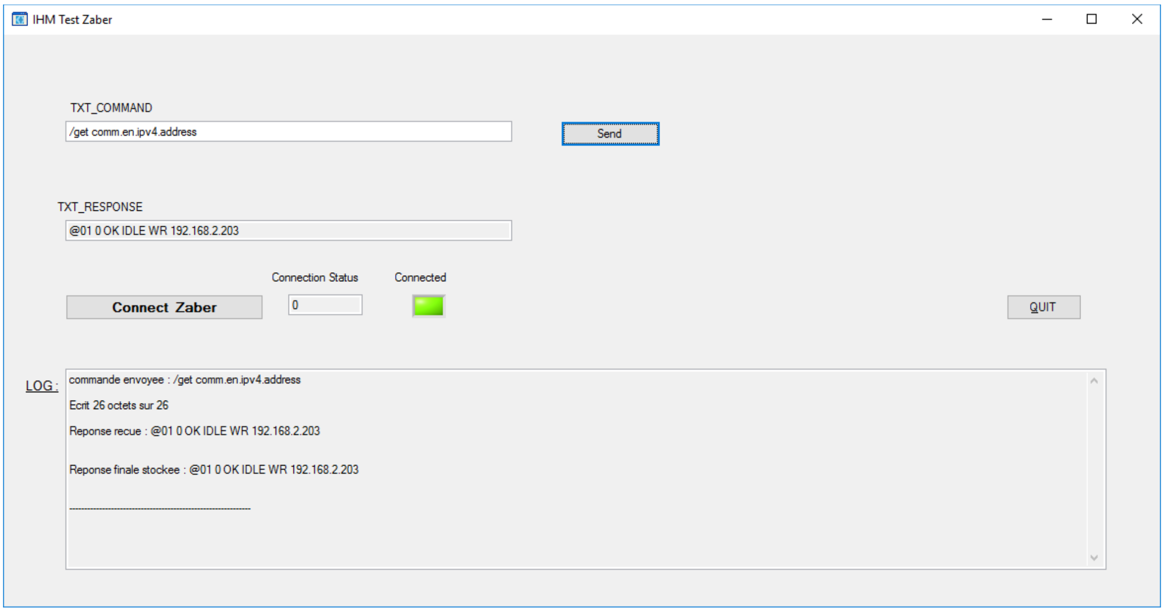


Figure 30: Zaber Debug HMI

2.4.1.1. TCP/IP Communication Architecture

The Zaber X-MCC controller communicates over Ethernet using a high-level, human-readable ASCII protocol transmitted via a TCP socket. This approach is compatible with embedded clients developed in C or C++ and facilitates integration into a multi-instrument control architecture based on TCP/IP.

2.4.1.2. ASCII Command/Response Format

To validate the communication layer and illustrate the ASCII command structure, several representative instructions were tested directly on the Zaber controller. Commands are sent as human-readable strings over the TCP/IP socket, and the controller replies with a status line confirming execution or reporting errors.

Each Zaber command follows a specific syntax (Fig. 31):

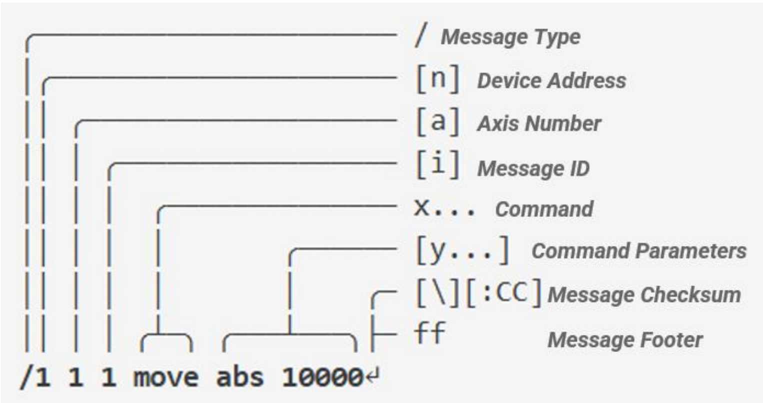


Figure 31 : ASCII Command Format

Here (Fig. 32) are examples of commands:

/1 1 get pos ↵	
/1 1 move abs 10000 ↵	/1 set maxspeed 100000 ↵
/2 1 move rel -12800 ↵	/1 get maxspeed ↵

Figure 32: ASCII positioning and configuration commands

- */1 1 get pos* : queries the current encoder position of axis 1 of device 1, expressed in microsteps.
- */1 1 move abs 10000* : moves axis 1 of device 1 to the absolute position corresponding to 10000 microsteps.
- */2 1 move rel -12800* : axis 1 of device 2 executes a relative move of −12800 microsteps from the current position.
- */1 set maxspeed 100000* : updates the maximum velocity to 100000 µsteps/s.
- */1 get maxspeed* : returns the current maximum velocity parameter (in µsteps/s).

During debugging, a typical test sequence involved querying the current position (/*get pos*), issuing an absolute move (/*move abs 1000*), and validating the final position by another *get pos* command. Similarly, configuration parameters such as maximum speed could be modified with *set* instructions and verified immediately with the corresponding *get*.

This set of examples demonstrates the simplicity of the ASCII protocol, while highlighting how low-level commands can be combined to build more complex motion sequences in the higher-level software layer.

Each command sent to the controller generates a structured ASCII response (Fig. 33), which provides the device address, axis identifier, command status, and any associated warning or error codes.

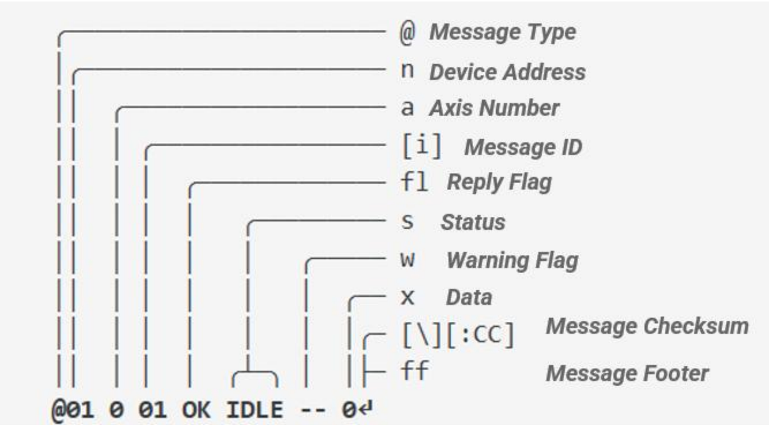


Figure 33: ASCII Response Format

2.4.1.3. Standalone Deployment on Offline Bench PC

After successful validation in the development environment, the Debug Human-Machine Interface (HMI) was deployed on the production computer associated with the spectral characterization bench. This target machine operates in a fully offline configuration, without internet connectivity or access to the LabWindows/CVI Integrated Development Environment (IDE). The objective of this deployment was to ensure that the debugging interface could function autonomously in the production environment while maintaining identical behavior to the development setup.

To achieve this, the software was compiled into a standalone executable using the LabWindows/CVI 2019 toolchain. The final build excluded debugging symbols, both to reduce the file size and to prevent the exposure of source code elements within the production environment. The compiled binary was designed to operate independently of the IDE, relying solely on the LabWindows runtime environment.

Since the target computer was isolated from the network, the installation of dependencies required a dedicated offline setup procedure. The LabWindows/CVI Runtime 2019 environment was installed using an installation package previously prepared on a connected machine. The installation process was performed entirely offline through the NI Package Manager, ensuring that all essential runtime components were available locally, including the TCP/IP communication libraries necessary for socket-based control of the Zaber motion controller. This approach ensured full compatibility with the isolated infrastructure of Sodern's optical benches, which are operated in cleanroom environments without external connectivity.

Once the runtime environment was in place, the executable was transferred to the target PC and executed under production conditions. Initial tests confirmed correct initialization of all software components and successful establishment of the TCP/IP connection with the Zaber X-MCC controller. The ASCII command exchanges between the HMI and the controller were monitored through the integrated log system, verifying proper bidirectional communication and acknowledgment of all motion commands.

The system demonstrated stable behavior identical to that observed in the development environment. Network communication remained continuous throughout extended test sessions, and no protocol errors or timeouts were detected. The standalone HMI thus proved capable of operating autonomously on the bench PC, providing Sodern operators with a reliable diagnostic and manual control interface without the need for the LabWindows IDE.

This deployment validated the portability and robustness of the software architecture. It demonstrated that the tool could be seamlessly integrated into Sodern's cleanroom computing infrastructure, fully compliant with the company's security and isolation standards.

2.4.1.4. Operational Benefits of the Debug HMI

The debug environment was first used for protocol verification, ensuring that the command syntax was correctly interpreted by the controller. It also provided real-time feedback, making it possible to directly observe the replies and troubleshoot configuration issues such as incorrect axis identifiers or parameter formatting. Finally, the tool served to validate network compatibility, confirming that the controller could be accessed from Sodern's offline measurement PC through the dedicated Ethernet network without encountering firewall restrictions or latency problems.

2.4.2. Spectral Bench Debug HMI

Following the initial controller-focused debug tool, a second HMI (Fig. 34) was developed to centralize communication with all instruments of the spectral characterization bench. Its purpose was to provide a single control point for validating both the performance of the new motion controller and its interoperability with the Photodiode Power Meter, the Optical Switch, and the Optical Spectrum Analyzer. The implementation of this HMI also leveraged and adapted methods previously developed by a former intern for communicating with the bench instruments, ensuring continuity with existing practices while extending them to support the new hardware configuration.

The HMI was structured into four dedicated sections, one per instrument:

- Status Indicators: green/red LEDs showing whether the instruments are connected.
- Command Input Fields: Text box to manually enter protocol commands.
- Send Buttons: Triggers the transmission of the command to the instrument.
- Log Area: Displays raw instrument responses and error messages.

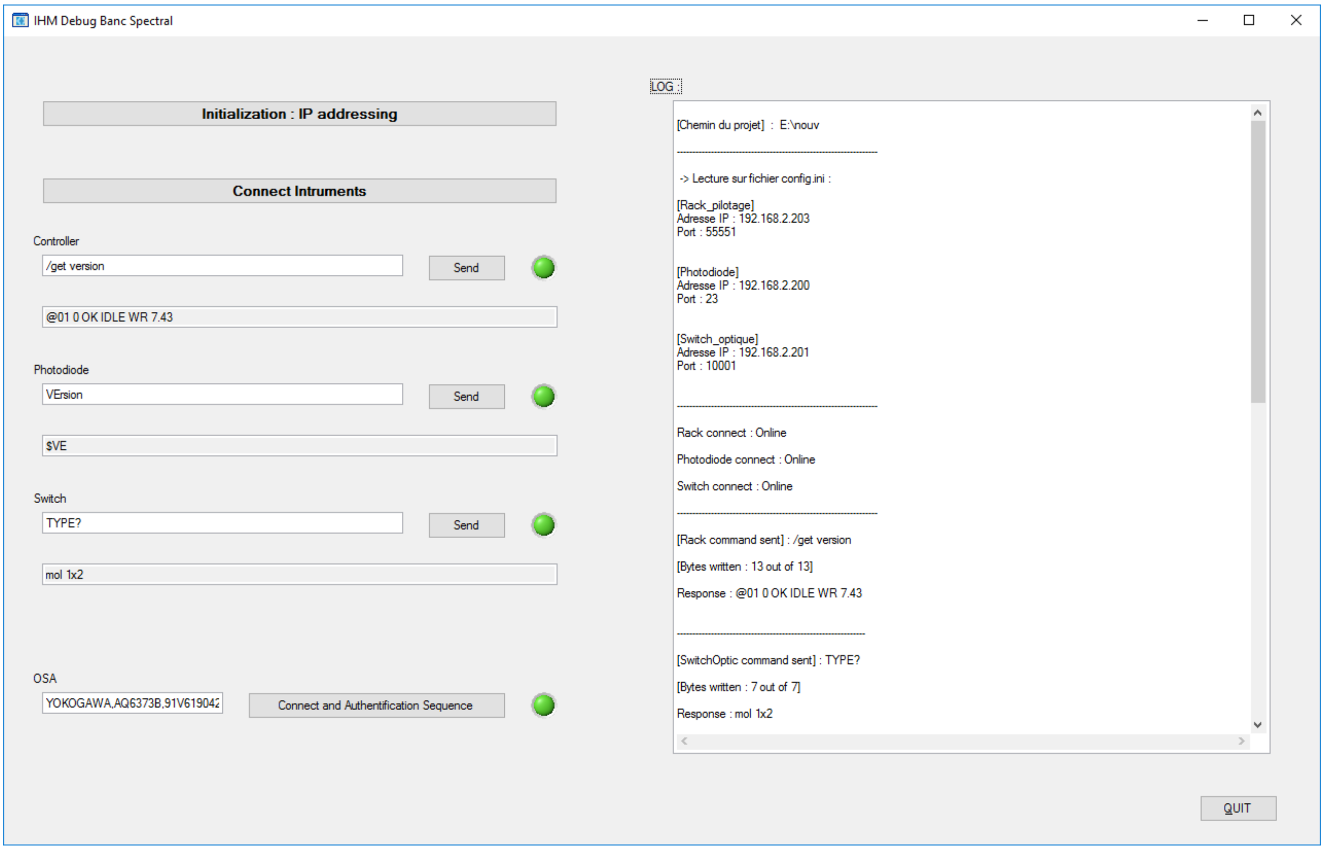


Figure 34: Spectral Bench Debug HMI

2.4.2.1. Config.ini file

In this HMI, the connection management system was built upon a software architecture initially developed by a former intern, which was specifically designed to retrieve and apply instrument parameters from an external configuration file (*config.ini*). This design allows network parameters (such as IP addresses and TCP ports) to be modified without recompiling the application, enabling quick reconfiguration of the bench and straightforward troubleshooting.

The process begins with the declaration of a static array of structures (one per instrument) each containing a character array for the IP address and an integer for the TCP port. This is defined in the communication module so that it is persistent and easily accessible during initialization.

At startup, the *ReadConfigFile()* function reads the *TCP_address* and *TCP_port* fields from the *.ini* file and stores them in the corresponding structure entries.

Finally, during the connection phase, these stored parameters are passed directly to the relevant *ClientTCPConnect()* function for each instrument, ensuring a uniform and easily reconfigurable connection process.

The complete implementation of Read_Config_File() spans too much lines, including error handling and macros. For readability, and for confidentiality reasons, only the simplified pseudo-code is presented in **Annex B.1.** showing the essential logic: reading parameters from config.ini, storing them in structured arrays, and using them to establish TCP connections.

2.4.2.2. Connection Management

Both Photodiode and Optical Switch devices use TCP sockets with fixed IPs and ports from the config file. The communication layer provides simple "*connect*" and "*send*" functions.

The OSA was integrated via VISA (Virtual Instrument Software Architecture: *viOpen*, *viPrintf*, *viScanf*, …), ensuring compatibility with the bench while allowing comparison with raw TCP. VISA abstracts socket handling for ease of use, whereas raw TCP requires manual protocol management but offers finer control.

At startup, the app reads the config.ini, loads IPs, and attempts connections. Successful links turn the status LED green. Then operators can send commands, with responses shown in the text box and log window for debugging.

Communication with each device was tested using representative commands, and successful responses confirmed full network compatibility with the bench PC. This second debug HMI proved essential for validating the "read–store–act" architecture based on the external config.ini file, enabling early multi-instrument integration tests, and providing a reusable modular framework for future expansion. It also serves as a functional reference for the final operational HMI to be integrated into the industrialized bench software.

2.4.3. Summary and Usefulness of the debugging tools

The debugging software tools developed during this project played a critical role in validating communication with the new motion controller prior to its full-scale integration into the spectral characterization bench. Beyond their immediate utility for system verification, these tools provide Sodern with a set of reusable and extensible utilities for future development and maintenance activities. Moreover, they established a scalable software framework suitable for subsequent industrial deployment. Both interfaces, along with their associated configuration files, commented source code, and deployment instructions, have been fully documented and archived to ensure long-term traceability and ease of reuse.

2.4.4. Implementation of the Automated Measurement Program

The development of the console-based background executable aimed primarily to assess the performance of the new motion controller in carrying out automated measurement sequences, with particular attention to execution time relative to the legacy control system. The implemented test sequence followed a simplified trajectory compatible with the prototype configuration, while remaining representative of a standard measurement operation on the full spectral characterization bench.

In addition to performance assessment, the program was designed to validate dynamic axis-to-connector assignment, demonstrate full axis configuration through an external .ini file—thereby enabling rapid system reconfiguration without manual modification—, and establish a minimal yet robust codebase suitable for future industrial control integration. Furthermore, continuous event logging to a structured file was implemented to ensure traceability, facilitate debugging, and support long-term maintainability.

2.4.4.1. Core Functions Used by the Final Executable

Two core functions ensure reliable interaction with the Zaber motion controller:

Send_zaber_command() : handles the transmission of ASCII commands over TCP and the collection of replies. It relies on a finite state machine to (i) send the command with timeout control, (ii) wait for incoming data without blocking, and (iii) return the full ASCII reply once available. Every step (command, response, errors) is logged, providing full traceability.

Decode_zaber_rep_cmd() : parses these ASCII replies into structured information. It extracts device, axis, command ID, status, warnings, and payload, then updates the internal model structure ("*axis_params*") accordingly. Status changes, encoder positions, and configuration data are all logged.

Together, these two functions form a robust loop: command → receive → decode → update → log. This mechanism guarantees synchronization between the software and the hardware and supports Sodern's requirement for dynamic axis reassignment.

2.4.4.2. Step 1 – Read the configuration file (config.ini)

Upon launch, the executable reads all required system parameters from the external configuration file (Fig. 35). The parameter configuration process relies on the same method as described for reading the config.ini file IP/Port settings; the pseudo-code of this routine is provided in **Annex B.1.** This file contains the network settings for the control rack, photodiode, optical switch, and OSA, as well as the mapping between each logical axis (X, Y, Z, theta, diaphragm) and its corresponding physical controller device and connector index. It also stores all axis parameters, including closed-loop tolerances, driver currents, microstepping resolution, encoder direction, homing speeds and so far…

A deliberate design choice was made to store the full set of parameters for every axis in the .ini file (approximately 150 parameters per axis). By rewriting all parameters at each execution, the system guarantees a known and repeatable configuration state. This method also removes any dependency on pre-existing settings stored in the controller, ensuring full independence from Zaber Launcher. In practice, this approach allows rapid recovery or reconfiguration in the event of hardware changes or controller replacement, while making it possible to dynamically reassign an axis to another connector simply by editing the *.ini* file.

```
[Axe_X]
connecteur = 1
init_speed = 403046
microstep_resolution = 0.00000813

peripheral.id = 88803

; Closed Loop
cloop.timeout = 100
cloop.displace.tolerance = 128
cloop.enable = 1
```

```
[01-08-2025 15:59:26] : [INFO] Command sent : /1 1 set cloop.displace.tolerance 128
[01-08-2025 15:59:26] : [INFO] Response received : @01 1 OK IDLE FO 0
[01-08-2025 15:59:26] : [INFO] Command sent : /1 1 set cloop.enable 1
[01-08-2025 15:59:26] : [INFO] Response received : @01 1 OK IDLE FO 0
```

Figure 35: Extract of config.ini file (left) and the console log showing loaded parameters (right)

2.4.4.3. Step 2 – Connect to the instruments

Once the configuration file has been fully parsed, the executable establishes network connections (Fig. 36) to all instruments specified in the .ini file. This includes opening TCP/IP sockets to the motion control rack, the photodiode, and the optical switch, as well as initiating a VISA session with the OSA. Each connection attempt is performed sequentially, with timeout management to avoid blocking the execution in case a device is powered off or unreachable. In the event of a failed connection, the program stops execution preventing any subsequent command from being sent.

```
[01-08-2025 15:59:21] : [INFO] Rack Online
[01-08-2025 15:59:21] : [INFO] Photodiode Online
[01-08-2025 15:59:21] : [INFO] Switch Online
```

Figure 36: Console output showing connection attempts and status

2.4.4.4. Step 3 – Setup the motion control rack

Using the configuration loaded in Step 1, the program sequentially configures all active axes (Fig. 37) on the Zaber controllers. This includes setting the driver parameters (run and hold currents, microstepping resolution, motor direction; see 2.3.2.3. Configuration of Main Parameters), applying closed-loop control settings (position tolerances, timeouts, stall detection parameters), configuring the encoder direction and feedback ratio, and defining software travel limits in microsteps according to the mechanical stroke of each axis. All parameters are written via the ASCII command protocol over TCP sockets.

To determine the exact sequence in which these parameters must be applied, a specific methodology was used with the Zaber Launcher software. An axis was configured from scratch, and the "*Apply*" button was clicked while simultaneously monitoring the integrated console output of Zaber Launcher. This revealed the precise ordering of configuration commands as sent by the official tool, as well as certain "hidden" intermediate commands required for proper axis initialization. The same sequence was then faithfully reproduced in the code, with the corresponding values retrieved from the .ini configuration file, ensuring that the automated setup behaves identically to a manual configuration through Zaber Launcher.

```
[01-08-2025 15:59:27] : [INFO] Command sent : /1 1 set cloop.timeout 100
[01-08-2025 15:59:28] : [INFO] Response received : @01 1 OK IDLE FO 0

[01-08-2025 15:59:28] : [INFO] Command sent : /1 1 set driver.current.approach 33
[01-08-2025 15:59:28] : [INFO] Response received : @01 1 OK IDLE FO 0

[01-08-2025 15:59:28] : [INFO] Command sent : /1 1 set driver.current.hold 10
[01-08-2025 15:59:28] : [INFO] Response received : @01 1 OK IDLE FO 0

[01-08-2025 15:59:28] : [INFO] Command sent : /1 1 set driver.current.run 33
[01-08-2025 15:59:28] : [INFO] Response received : @01 1 OK IDLE FO 0
```

Figure 37: Code snippet from executable sending axis X configuration commands (extract)

2.4.4.5. Step 4 – Setup photodiode, optical switch, and optical spectrum analyzer

Each instrument in the measurement chain is initialized to a known, ready state to ensure consistent starting conditions before the automated sequence begins. The photodiode is configured to its default measurement range, the optical switch is positioned on a predefined channel corresponding to the selected optical path, and the Optical Spectrum Analyzer (OSA) is set to the appropriate wavelength span, resolution, and averaging time required for the test scenario.

This procedure guarantees that all devices are synchronized with the measurement sequence without requiring manual intervention from the operator. The initialization approach for these instruments was derived from the work of a previous intern, who had extensively studied the bench's operational behaviour with the legacy control rack. By reusing and adapting these validated sequences, the program ensures compatibility with existing measurement practices while maintaining consistency in data acquisition parameters.

2.4.4.6. Step 5 – Initialize the rack

The initialization sequence ensures that each active axis is aligned to a consistent and absolute reference before the measurement campaign starts. This reference corresponds to the first encoder Z mark encountered when moving from the positive end of travel. The entire procedure is automated and executed in parallel for all axes to minimize setup time.

The full C implementation of the axis initialization routine is provided in **Annex B.2**. (for clarity, only the simplified pseudo-code is presented).

At the beginning of the sequence, closed-loop regulation is explicitly disabled (*cloop.enable = 0*) to avoid potential stall during homing moves, and the minimum travel limit is temporarily set to an extremely high value (*lim.max = 100000000*). This ensures that the homing routine can reach the limit sensor regardless of the initial axis position.

The process then follows three main stages. First, each axis is driven at constant speed in the positive direction, moving away from the negative limit to secure a valid starting position. Every 100 ms, the program queries the motion state (*IDLE* or *BUSY*) to detect when the axis activates its limit sensor. This polling runs in parallel for all axes.

Once the sensor is reached, the program issues the homing command (Fig. 38) (homing is always executed in the negative direction, that's why each axis was driven at constant speed in the positive direction, which guarantees that no limit switch was encountered during homing commands). The axis continues until the encoder detects the first Z reference mark from the positive side. During this movement, the state is again checked at 100 ms intervals to capture the exact detection event.

When all axes have reached their respective encoder Z reference marks, the initialization is considered complete. At this stage, closed-loop control is re-enabled (*cloop.enable = 1*) and the travel limits are restored to their nominal values, ensuring safe operation for subsequent measurements.

It must be noted that the logical signs used in software are inverted compared to the physical motion of the axes. This convention, defined during the direction-mapping study described in section *2.3.2.5. Limit and Direction Configuration*, is internally managed by the initialization routine so that operator commands remain intuitive.

```
if (status_flags[i_i] == 0)
{
    // Envoyer commande HOME
    memset(tc_b, 0, BUFFER_TCP_IP_SIZE);
    sprintf(cCmd, ZABER_CMD_HOME, device, axe);
    E_CL(Send_zaber_command(cCmd, tc_b, pc_err), i_nerr);

    memset(tc_b, 0, BUFFER_TCP_IP_SIZE); \
    sprintf(cCmd, ZABER_SET_LIMIT_MAX, device, axe, axe_params[i_i].limit_max);
    E_CL(Send_zaber_command(cCmd, tc_b, pc_err), i_nerr);

    homed_flags[i_i] = 1;
    sprintf(tc_b, "Axe %d : end reached -> Homing start", i_i+1);
    Add_Message_Log(e_Info, tc_b);
}
```

Figure 38: Homing code loop

2.4.4.7. Step 6 – Execute the predefined measurement trajectory

The goal of this step was to reproduce, on the prototype axis, a simplified yet representative measurement cycle of the spectral bench. The sample holder traveled 40 mm in 50 µm steps (800 points); at each position, encoder feedback was logged, simulating synchronization with future spectral acquisitions. Trajectory parameters (speed, acceleration, tolerances) were loaded from the config file, ensuring reproducibility and avoiding manual setup. The OSA call was kept in the code but disabled to focus on motion and logging performance.

The full C implementation of this measurement sequence is provided in **Annex B.3.** (for confidentiality, only the simplified pseudo-code is presented).

The loop logic followed four phases: (1) Send Absolute Move – target position converted to microsteps; commands sent in ASCII over TCP/IP. (2) Wait for Completion – the program polled every 100 ms until IDLE, while logging BUSY/WS warnings. (3) Read Encoder Position – actual feedback retrieved with get pos, converted back to mm to check consistency. (4) Log Data – index, target, encoder value, and raw exchanges recorded for traceability.

In the final setup, each step will also trigger OSA acquisition; the function is already implemented but was disabled here.

Engineering considerations: The use of 50 µm increments offered a good compromise between resolution and total test duration: with 800 iterations, the trajectory (Fig. 39) was long enough to validate the stability of communication and control loops while remaining executable in a few minutes. Synchronizing data acquisition with the IDLE state ensured that measurements would always be taken under stable mechanical conditions. The exhaustive logging of commands, responses, and encoder values provided a detailed trace for performance analysis, including the possibility to detect cumulative drifts or transient anomalies.

```c
//BOUCLE PERMETTANT DE TESTER TOUS LES 50um SUR 40mm AVEC MESURE DE L'ANALYSEUR EN //
int i_iteration = 0;
start = clock();
for(float i_position = 0.0; i_position <= 40.00 ; i_position += 0.05)
{
    //FONCTION DE PILOTAGE DE LA POSITION SUR L'AXE EN FONCTION DE LA POSITION EN MM
    E_CL(Zaber_cmd_position_abs(AXE_X, i_position, tc_err), i_nerr);

    //FONCTION D'ARRET DU MOTEUR
    E_CL(Wait_Axis_Idle(AXE_X, tc_err), i_nerr);
    E_CL(Zaber_cmd_get_position(AXE_X, tc_err), i_nerr);
    sprintf(tc_b, "Mesure n_%d : Position Encodeur (mm): %f\n",i_iteration, axe_params[AXE_X].PosAxe_mm);
    Add_Message_Log(e_Info, tc_b);

    //FONCTION PERMETTANT DE LANCER UNE MESURE VIA L'ANALYSEUR ET D'IMPLEMENTER LES DONNEES DANS LE FICHIER DE MESURE
    //GetTraceFromAnalyseur(tc_err, sharedData.osaSession, sharedData.rmSession, fichier_resultat_mesure);
    i_iteration++;

}
```

Figure 39: Code snippet of the measurement loop

2.4.4.8. Structured Log File

To ensure complete traceability and reproducibility of automated measurement sequences, the developed application implements a structured logging system that records all key execution events in a timestamped log file (Fig. 40). This log mechanism, fully integrated into the software architecture, provides both real-time supervision during operation and post-run traceability for analysis and diagnostics.

```
-------------------------
[01-08-2025 15:59:21] : [INFO] >> Ouverture de l'application par PC sur ***************
[01-08-2025 15:59:21] : [INFO] End of the config.ini file reading
[01-08-2025 15:59:21] : [INFO] Rack Online
[01-08-2025 15:59:21] : [INFO] Photodiode Online
[01-08-2025 15:59:21] : [INFO] Switch Online

[01-08-2025 15:59:21] : [INFO] Command sent : /1 1 set peripheral.id 0
[01-08-2025 15:59:21] : [INFO] Response received : @01 1 OK IDLE -- 0

[01-08-2025 15:59:21] : [INFO] Command sent : /1 1 set peripheral.id 88803
[01-08-2025 15:59:22] : [INFO] Response received : @01 1 OK IDLE WR 0
```

Figure 40: Extract of the generated log file

At program startup, the function *Setup_Log_File()* automatically creates a new log file within the project's */log* directory. If a log file with the same application name already exists from a previous day, it is automatically renamed with the corresponding execution date, following the convention *<application>_YYYY-MM-DD.log*. This mechanism prevents data loss due to overwriting while maintaining chronological storage of historical logs. For the active session, the log file always retains the fixed name *<application>.log*, allowing operators and diagnostic tools to access current data without manual intervention.

Each log entry begins with a timestamp accurate to the second, formatted as *[DD-MM-YYYY HH:MM:SS]*, followed by a severity level identifier:

- INFO for normal operations such as initialization, connection, or parameter application,
- WARN for non-blocking anomalies such as temporary communication delays,
- ERROR for critical faults leading to process interruption.

For example, an entry such as

[01-08-2025 14:28:53] : [INFO] Rack Online

indicates a successful connection to the rack controller at the given time.

Throughout execution, the program's various modules use the *Add_Message_Log()* function to record major operational events. These include application startup, parsing of configuration files, instrument connection results, and the application of motion parameters through ASCII commands. During operation, every motion instruction and response is logged sequentially, ensuring a complete command/response trace between the supervisory software and the motion controller. An extract from a recorded trajectory, for instance, includes entries such as:

[01-08-2025 14:34:06] : [INFO] Cmd abs :/1 1 5 move abs -12300
[01-08-2025 14:34:06] : [INFO] Response received : @01 1 05 OK BUSY WS 0

Such entries enable the reconstruction of all low-level interactions, allowing engineers to diagnose potential synchronization or communication issues precisely.

At the end of each automated measurement sequence, the software calculates the total execution time of the operation and appends it to the log. This provides a direct performance metric that can be compared to cycle times obtained using the legacy control system, offering a quantitative means of assessing the efficiency of the new architecture.

The structured logging framework provides multiple advantages. It guarantees traceability, ensuring that every event, from initialization to shutdown, is permanently recorded for later review. It enhances debugging efficiency, allowing developers to reconstruct the exact order of commands and system responses in case of anomalies. It also supports performance benchmarking, facilitating the objective comparison of control strategies across different software versions or hardware configurations. Finally, its standardized and documented format ensures compliance with Sodern's cleanroom operational procedures, allowing logs to be archived directly within the company's technical documentation system.

The log file thus serves not only as a diagnostic tool but also as an essential component of the software's validation and verification process, ensuring that every automated sequence on the spectral characterization bench can be analyzed, reproduced, and audited in full accordance with aerospace-grade quality standards.

2.4.4.9. Dynamic Axis Reassignment Based on Controller Connector Mapping

A key requirement from Sodern was to automate axis configuration in C++ so that any logical axis could be reassigned to a spare connector in case of hardware failure. In the final setup, two Zaber X-MCC controllers will drive five active axes, with one spare connector acting as a hot-swappable backup. If a channel fails, the motor can simply be re-plugged into the spare connector, and the software automatically restores its configuration without requiring Zaber Launcher Software.

This capability was implemented by extending the `config.ini` file to include both connector mappings and complete axis configuration parameters. Each logical axis (e.g., X, Y, Z) points to a connector entry that specifies the associated device ID and axis index. The configuration file also stores all operational parameters (closed-loop tolerances, driver currents, microstepping resolution, peripheral ID, etc.), ensuring that the setup can be reapplied in full.

During initialization, the *Setup_Rack()* procedure calls *MapAxeToConnector()* to resolve each logical axis to its assigned connector. Invalid mappings are caught early, while valid axes are configured with the exact same parameter sequence as in Zaber Launcher. At runtime and after a *Decode_Zaber_Response()* function call, the *Find_Axis()* function ensures that responses from the controllers are always associated with the correct logical axis, even after a reassignment.

In practice, if a connector fails, the operator only needs to move the cable and update the connector number in `config.ini`. On restart, the software re-applies the complete configuration and resumes normal operation. This minimizes downtime and preserves configuration fidelity.

2.4.4.10. Final Results of the Automated Measurement Program

The final build of the executable successfully executed the complete initialization and measurement sequence as designed. Upon startup, the setup of all motion axes was carried out correctly, with each logical axis mapped to the appropriate physical connector according to the configuration specified in the external .ini file. All initialization routines completed without errors, confirming that communication with every instrument was established and stable.

The structured log file generated during execution provided a clear and detailed record of the process, including axis setup parameters, initialization steps, motion commands, photodiode readings, and any status changes. This level of traceability ensures that the measurement sequence can be fully reconstructed for debugging or validation purposes.

The predefined measurement trajectory was executed from start to finish in 1617 seconds or **29 minutes** (Fig. 41), covering all programmed positions with consistent motion control and synchronized data acquisition. The absence of communication faults or execution errors validated both the robustness of the control logic and the reliability of the integrated hardware–software architecture.

```
[01-08-2025 15:00:59] : [INFO] Command sent : /1 1 4 get pos
[01-08-2025 15:00:59] : [INFO] Response received : @01 1 04 OK IDLE WS -4920009

[01-08-2025 15:00:59] : [INFO] get position axe(1), etat(IDLE)
[01-08-2025 15:00:59] : [INFO] Mesure n_800 : Position Encodeur (mm):-40.000072

[01-08-2025 15:00:59] : [INFO] Duree : 1617.462000 s
```

Figure 41: Execution time of the Automated Measurement Program

3. Project Assessment and Reflection

This chapter presents a critical analysis of the outcomes of the motion control system project and reflects on the management and execution aspects of the work. Building upon the technical implementation and validation described in Chapter 2, it provides an assessment of the system's performance, identifies key lessons learned during the project, and examines the organizational strategies and challenges encountered. The insights presented here are intended to inform both the further development of the spectral characterization bench and the management of similar engineering projects in an industrial context.

3.1. Evaluation and Lessons Learned

3.1.1. Performance with Respect to Specifications

The upgraded motion-control subsystem fulfills the functional and integration requirements established at project start. All active axes operated reliably with the existing bipolar stepper motors and RS-422 incremental encoders, preserving full compatibility with the bench's legacy SUB-D connectors and cleanroom constraints (ISO 6). Travel-limit detection via magnetic reed sensors was interpreted deterministically by the controller, enabling safe homing and enforcement of motion boundaries.

The TCP/IP communication stack proved stable during extended operation; the controller responded consistently to both low-level ASCII commands and HMI-level procedures. Integration within LabWindows/CVI enabled seamless coordination with the photodiode, optical switch, and optical spectrum analyzer; the structured logging facility provided second-level time-stamping and end-to-end traceability.

Critically, a representative acquisition sequence executed in **29 min—versus nearly 5 h with the legacy rack—** demonstrating a step-change in throughput without compromising positioning accuracy or environmental constraints. Overall, the new architecture not only replaces the obsolete system but exceeds Sodern's expectations for reliability, maintainability, and operational efficiency.

3.1.2. Identified Limitations and Technical Bottlenecks

Several constraints emerged during validation. First, the controller's digital outputs cannot directly drive inductive loads at the required current; the adopted solution (interposing relays with flyback protection) is robust but increases wiring complexity and maintenance. A future revision should evaluate solid-state drivers or dedicated 24 V output modules.

Second, absolute referencing relies on encoder Z-index detection; given multiple Z marks on encoder scales, unambiguous homing requires a constrained approach direction and positive confirmation of the selected Z event (status polling and distance windowing) prior to zeroing.

Third, closed-loop behavior is functional but would benefit from axis-specific tuning (settling tolerance, stall thresholds) under worst-case load. Timeouts and polling cadences were set empirically to balance responsiveness and robustness; a systematic optimization (e.g., controller-side notification or adaptive polling) could reduce idle latency.

Finally, the external config.ini stores raw integers (microsteps, counts, limits) without explicit units; while efficient to parse, it is error-prone. A schema with units and bounds checking would improve safety and maintainability.

3.1.3. Reusability and Handover

Reusability guided both hardware and software design. The prototype rack uses standardized industrial components (Zaber X-MCC controller, differential encoder inputs, open-collector outputs, solid-state relays); cabling follows labeled SUB-D/Phoenix conventions with complete pinout documentation, facilitating replication to the full five-axis configuration.

On the software side, complete axis profiles were authored in Zaber Launcher and exported; ~150 parameters per axis are loaded at runtime from an external .ini, then validated and applied, eliminating recompilation during bench reconfiguration. Two tools were delivered: a modular debug HMI for interactive tests and a console executable for automated initialization/measurement with structured logging. All source code, wiring diagrams, configuration files, test reports, and deployment instructions were archived in Sodern's internal system with versioning and traceability to enable seamless continuation by future teams.

3.1.4. Lessons Learned

Three insights stand out. (i) Hardware–software co-design is decisive: anticipating output-stage limits enabled early definition of relay interfaces and protection, avoiding late-stage redesigns. (ii) Documentation is a multiplier: the absence of legacy references incurred reverse-engineering overhead, while the new system's comprehensive schematics, pinouts, and parameter sets now de-risk maintenance and upgrades. (iii) Designing for offline deployment—from dependency packaging to runtime diagnostics—shortened the path from prototype to cleanroom operation.

3.1.5. Readiness for Industrialization

Despite its prototypical scope, the system is mature for integration into the operational bench. Electrical and mechanical compatibility have been validated; the software architecture is inherently scalable to additional axes and instruments; safety and communication mechanisms align with Sodern's operational standards and ISO-6 constraints. The next milestone is a production-grade HMI (LabWindows/CVI) consolidating the validated logic and enabling full five-axis operation. With this deliverable and minor refinements, the solution is ready for industrial deployment.

3.2. Project Management Reflection

3.2.1. Planning and Structuring the Work

The project was structured into three main phases to ensure a coherent progression from analysis to operational validation.

The first phase consisted of system analysis and conceptual design. It involved the detailed study of the legacy control architecture, identification of obsolescence issues, definition of functional requirements, and compatibility assessment with the new control hardware. This analytical stage established a clear baseline and technical roadmap for the redesign effort.

The second phase focused on prototyping and integration. A reduced-scale two-axis setup was implemented to validate the electrical interfaces and power distribution architecture prior to full deployment. Cable harnesses were custom-designed to comply with Sodern's cleanroom standards, while each subsystem — including encoders, sensors, and actuators — was tested individually to verify electrical integrity and signal conditioning.

Finally, the third phase addressed software development and deployment. This included the creation of dedicated debugging tools, development of the TCP communication layer, and the implementation of a standalone runtime application on the offline control workstation. The overall workflow followed an incremental validation strategy, where each milestone was confirmed through test results before advancing to the next. A continuous project log was maintained, recording configuration changes, troubleshooting actions, and technical decisions to ensure full traceability.

3.2.2. Autonomy and Collaboration

The project required a high level of autonomy in the daily management of tasks and problem-solving activities. While the design and implementation stages were carried out independently, periodic technical reviews with the supervisor ensured alignment with Sodern's engineering methodology. These reviews covered critical aspects such as component selection, encoder logic validation, and the assessment of electrical risks during integration.

Collaboration with Sodern's electronics and software teams provided targeted technical support, particularly during the verification of signal wiring and the configuration of runtime environments within secured networks. This hybrid mode — combining personal initiative with collective expertise — proved effective in maintaining both agility and compliance with the company's stringent operational procedures.

3.2.3. Communication and Reporting

Effective communication played a crucial role in maintaining project transparency and ensuring the continuity of work. Weekly progress briefings were organized to present key results, identify blocking points, and update short-term objectives. In parallel, all project artifacts (electrical schematics, configuration files, test data, and screenshots) were systematically stored in a shared repository accessible to the R&D team.

A comprehensive technical note was compiled at the end of the work, consolidating system compatibility analyses, wiring documentation, and experimental results. This document was designed to serve as a reference for future developments and to facilitate onboarding of new engineers who will extend the work toward full five-axis industrial integration.

3.2.4. Reflection

The project provided hands-on exposure to aerospace-grade constraints and to the full V-cycle from requirements to validation. It reinforced modular design for risk control, the value of prototypes for early integration feedback, and the centrality of clear documentation for maintainability and knowledge transfer. This project provided significant exposure to the aerospace engineering environment, characterized by high reliability requirements, traceability obligations, and multidisciplinary collaboration. Beyond the technical achievements, it also offered practical insight into the complete V-model development cycle, from initial requirements definition to final validation.

Several methodological principles emerged as key takeaways:

- the importance of modular and incremental design for risk mitigation,
- the value of early prototyping to detect integration challenges,
- and the necessity of structured documentation for long-term maintainability and knowledge transfer.

Overall, this experience strengthened both the technical and organizational competencies required of a mechanical and systems engineer, reinforcing the capacity to conduct complex engineering projects within industrial-grade constraints.

Conclusion and Future Perspectives

Summary of Achievements

This project led to the successful design, implementation, and partial deployment of a modern motion control system for Sodern's spectral characterization test bench, replacing an obsolete and undocumented setup. The work spanned the full engineering workflow, from requirement analysis and hardware integration to cable manufacturing, controller configuration, software development, and functional validation in a cleanroom environment.

Key accomplishments include the selection and validation of the Zaber X-MCC industrial motion controller, fully compatible with the existing stepper motors, encoders, and limit-switch logic of the bench. A complete interface cabling solution was designed and hand-assembled, integrating the necessary protection components.

A two-axis prototype was built and demonstrated, validating precise positioning, sensor feedback processing, and actuator triggering. Two modular software interfaces (HMIs) were developed: one dedicated to low-level debugging of the controller, and a second capable of centralizing the control of all bench instruments. The system was deployed on an offline secure test environment, with all runtime dependencies configured for stand-alone execution. Comprehensive documentation was delivered, covering wiring diagrams, compatibility adaptations, communication protocols, and deployment instructions, ensuring seamless handover to future engineering teams.

This combined hardware-software foundation is now technically ready for scaling to the full five-axis implementation required for the operational bench.

Limitations and Remaining Work

While the prototype phase successfully validated most critical functions, several items remain outside the scope of this project:

- Final integration and testing of all five motion axes in the operational rack.
- Execution and optimization of full measurement trajectories through the HMI.
- Integration of additional instruments (photodiodes, spectrometer, camera) into the unified control interface.
- Extended testing of error handling for scenarios such as communication loss, encoder signal anomalies, or actuator failures.
- Migration of the legacy operational complete software from LabVIEW to a LabWindows/CVI-based interface, leveraging the modular communication and configuration architecture developed during this project as a foundation for the new implementation.

Many of these tasks have already been prepared technically: interface logic exists for additional axes, and the software contains placeholders for future instruments. The migration effort in particular will benefit from the structured, reusable code developed here. As such, these remaining steps can be completed efficiently in a follow-up internship or engineering project.

Personal and Technical Lessons Learned

This project provided valuable exposure to space-grade engineering practices and offered a realistic view of system development within an industrial context. It reinforced the importance of anticipating hardware–software mismatches early and addressing them through staged, isolated integration steps. The use of modular prototyping proved highly effective for containing risk, accelerating debugging, and simplifying validation. The work also provided hands-on expertise in implementing TCP/IP-based motion control protocols within a constrained and secure environment. In addition, it highlighted the critical role of comprehensive documentation in ensuring traceability, maintainability, and effective knowledge transfer. Finally, it deepened the understanding of aerospace cleanroom constraints and their direct impact on hardware design, assembly, and testing procedures.

Conclusion

This six-month final project at Sodern-ArianeGroup marked the culmination of my training as a mechatronics engineer, integrating the multidisciplinary knowledge developed at ENSIL-ENSCI with the advanced technical and methodological competencies acquired at the Politecnico di Torino. The project offered an exceptional opportunity to contribute to a concrete, high-stakes engineering challenge situated at the crossroads of precision mechanics, electronics, and software integration — a combination emblematic of modern mechatronic systems within the demanding context of the aerospace industry.

The work began with a thorough system analysis of the legacy motion control subsystem used in Sodern's spectral characterization bench. This stage enabled the identification of obsolescence factors and functional limitations, as well as the definition of a clear roadmap for modernization. Building upon this foundation, a new modular and industrial-grade control architecture was designed and implemented. The contributions encompassed the selection and validation of a modern motion controller, the electrical design and assembly of a prototype control rack compliant with technical and safety requirements, and the development of dedicated software tools under LabWindows/CVI to ensure reliable communication, closed-loop control, and automated measurement sequences.

The resulting system successfully demonstrated functional equivalence and improved maintainability compared to the obsolete configuration, while meeting the constraints imposed by cleanroom operation and aerospace qualification standards. Furthermore, the Technical Specifications document for the Motor Control Drawer was drafted to formalize the hardware and software configurations, providing a solid foundation for future procurement, industrial deployment, and system certification.

Beyond the tangible technical outcomes, this internship represented a valuable exercise in engineering methodology. The project required the application of modular design principles to facilitate scalability and risk control, the establishment of systematic testing procedures to guarantee functional robustness, and the production of comprehensive documentation to ensure maintainability and knowledge transfer within the company. These practices, embedded in Sodern's industrial culture, provided first-hand experience of the V-cycle development process and of the rigorous standards governing aerospace systems engineering.

From a personal and professional standpoint, this experience has been a decisive step in my formation as a systems-oriented engineer. It reinforced my ability to approach complex problems through structured analysis and cross-disciplinary reasoning, while highlighting the importance of teamwork, traceability, and long-term design thinking in industrial environments.

Ultimately, this thesis confirmed my aspiration to pursue a career dedicated to the development of high-reliability, multidisciplinary systems for advanced technological applications. I express my sincere gratitude to Sodern, ENSIL-ENSCI and the Politecnico di Torino, for providing the technical environment, guidance, and trust that made this project possible, and for allowing me to bridge the gap between academic learning and industrial innovation.

Bibliography

[1] Airbus Defence and Space, *Optical Design, Challenges and Development of the METimage Instrument*, Airbus DS Publication, 2017.

[2] H. Thiele and M. Gisi, “Multispectral Filter Spectral Characterisation Setup”, *Advances in Optical Thin Films VII*, Proc. SPIE 10691, 2018.

[3] R. Le Goff and E. Prévost, “Bandpass Filters for Multispectral Remote Sensing Imagers”, *Proceedings of the 68th International Astronautical Congress (IAC)*, IAC-17, 2017.

[4] ISP*, Dossier de Définition du Banc Spectral*, Internal Technical Report ISP-19A1007NT, Sodern, 2021.

[5] R. Bruneau, *Amélioration et Portage de Logiciel de Banc de Test*, Sodern Internal Report, 2024.

[6] Zaber Technologies Inc., *X-MCC Series Controller User Manual*, Version 1.2, 2023.

[7] Thorlabs Inc., *Kinesis® Motion Control System – Software and APIs: Technical Documentation*, 2022.

[8] National Instruments, *LabWindows™/CVI Developer Guide*, NI Developer Zone, 2021.

Glossary

HMI (Human-Machine Interface)	Graphical or console-based interface allowing an operator to control and monitor a system.
TCP/IP	Transmission Control Protocol / Internet Protocol – communication protocol used for data exchange over Ethernet.
.ini file	Text-based configuration file containing key-value pairs used to set up parameters.
Closed-loop control	Control method using feedback from sensors (e.g., encoders) to adjust commands and correct errors in real-time.
End-of-travel sensor	Device that detects when a moving axis reaches its mechanical limit.
Incremental encoder	Position sensor providing relative displacement information and reference pulses.
Stepper motor	Electric motor that moves in discrete steps, enabling precise positioning.
Z-signal reference mark	Reference pulse in incremental encoders used for homing or calibration.
EMI	Electromagnetic Interference
OSA	Optical Spectral Analyser
VISA	Virtual Instrument Software Architecture
IDE	Integrated Development Environment
ESD	Electrostatic Discharge
PPE	Personal Protective Equipment

List of Figures

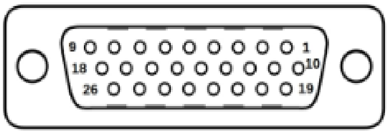
Appendices A – Electrical Schematics and Interfaces

A.1. Pinout mapping of connectors

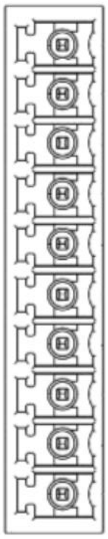
Embase SUBD 15 M	Codeur X	
Affectation des pins	Pin	Fonctions
	01	Alimentation codeur +5V
	02	Entrée différentielle A codeur incrémental
	03	Entrée différentielle B codeur incrémental
	04	Entrée différentielle Z codeur incrémental
	05	Entrée différentielle DATA codeur ENDAT 2.x
	06	Entrée différentielle CLOCK codeur ENDAT 2.x
	07	NC
	08	Alimentation codeur +5V
	09	Alimentation codeur 0V
	10	Entrée différentielle A/ codeur incrémental
	11	Entrée différentielle B/ codeur incrémental
	12	Entrée différentielle Z/ codeur incrémental
	13	Entrée différentielle DATA/ codeur ENDAT 2.x
	14	Entrée différentielle CLOCK/ codeur ENDAT 2.x
	15	Alimentation codeur 0V

Embase SUBD 15 F	Moteur X	
Affectation des pins	Pin	Fonctions
	01	Moteur Phase A
	02	Moteur Phase B
	03	0V
	04	Entrée tout ou rien n°1
	05	Entrée tout ou rien n°2
	06	0V sortie tout ou rien
	07	24V fins de course
	08	Moteur Phase A/
	09	Moteur Phase B/
	10	Sortie tout ou rien
	11	24V
	12	Capteur fin de course +
	13	Capteur fin de course -
	14	Capteur d'origine
	15	0V - GND

Bench connectors and their associated pins



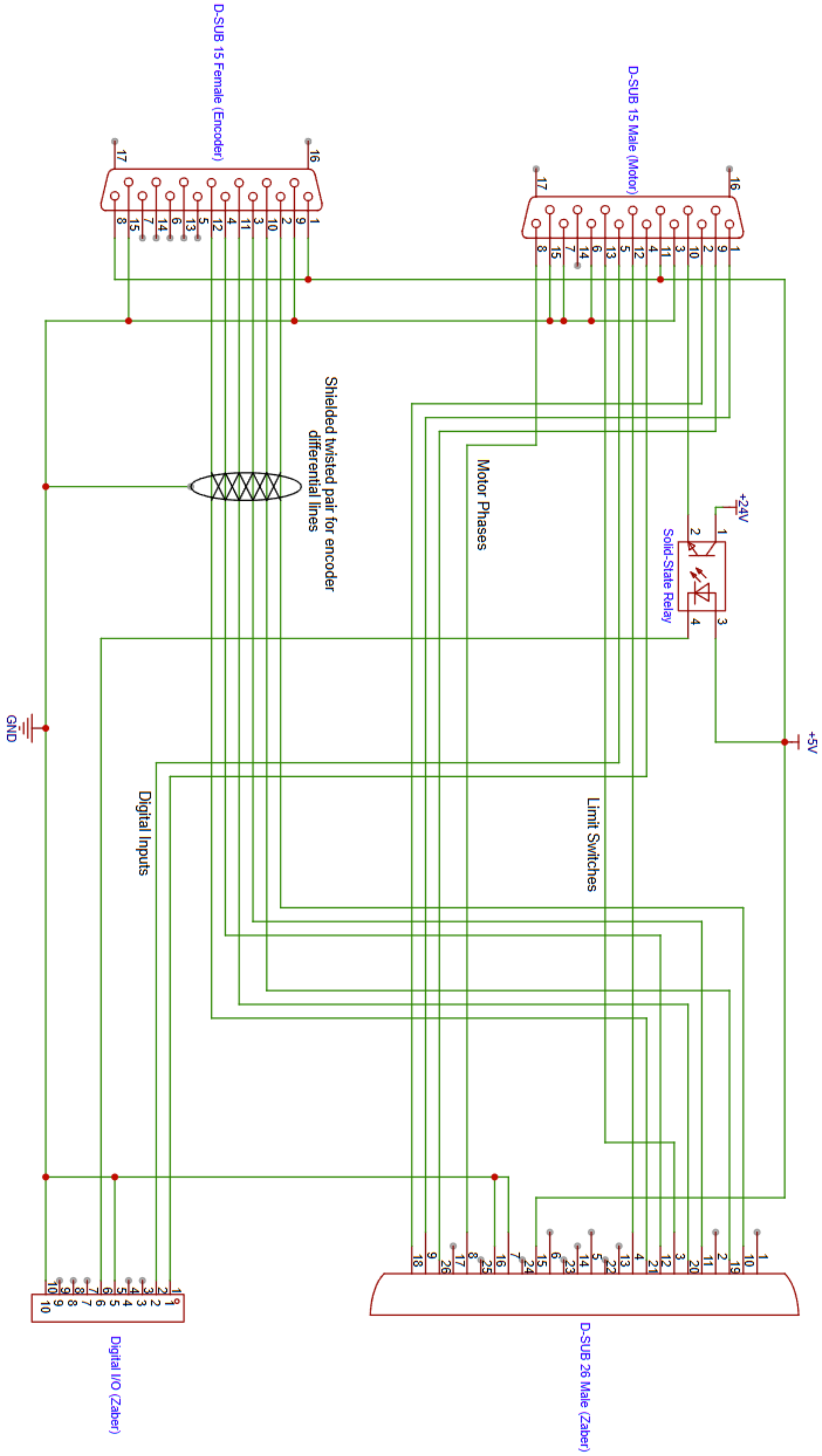
Pin	Description
1	AutoDetect Clock
2	AutoDetect Data
3	C Limit Sensor
4	Away Limit Sensor
5	Home Limit Sensor
6	Motor Over-Temperature
7	Ground
8	Motor A2 / Motor V
9	Motor A1 / Motor U
10	Differential Encoder A+
11	Differential Encoder B+
12	Differential Encoder Index+
13	Differential Encoder Error
14	Single-ended Encoder Index
15	+5 V
16	Ground
17	Brake-
18	Motor B1 / Motor W
19	Differential Encoder A-
20	Differential Encoder B-
21	Differential Encoder Index-
22	Single-ended Encoder A
23	Single-ended Encoder B
24	Reserved
25	Brake+
26	Motor B2



Pin	Description
1	Digital In 1
2	Digital In 2
3	Digital In 3
4	Digital In 4
5	Digital In Common
6	Digital Out 1
7	Digital Out 2
8	Digital Out 3
9	Digital Out 4
10	Digital Out Common

Zaber X-MCC connectors and their associated pins

A.2. Electrical Diagram of the one-axis cable



Electrical diagram of one-axis cable connecting the bench to the new Zaber controller (one per axis)

Appendices B – Software Implementation

B.1. Pseudo-code: Config.ini file management

```
//*---------------------------------------------------------------------------
//* Project : Spectral Bench Modernization
//*---------------------------------------------------------------------------
//* File Name : communication.c
//* Object : simplified pseudo-code for config file management
//* Creation : Intern_Mechatronics : Enzo MINACCIATTI Aug/2025
//*---------------------------------------------------------------------------
// NOTE: Full C code includes error macros, buffers, and parsing; here we keep the core
logic only.


// Include standard headers
#include <stdio.h>
#include <string.h>


// Global structure for instrument parameters
typedef struct {
    char ip_address;     // instrument IP
    int tcp_port;            // instrument port
} InstrumentConfig;


// Static array for all devices (e.g., photodiode, optical switch, rack)
InstrumentConfig AxisParameter[NB_DEVICES];




//*-------------------------------------------------------------------------------
//* Function Name : Read_Config_File
//* Object : read IP and port from config.ini and store them in AxisParameter[]
//* Input Parameters : char *pc_err // error message buffer
//* Output Parameters : int // error code (0 if OK, -1 if failed)
//*-------------------------------------------------------------------------------
int Read_Config_File(char *pc_err)
{
    // 1. Build full path to config.ini
    path = "<project_dir>/ini/config.ini";

    // 2. Open the INI file
    ini = Ini_Open(path);

    // 3. Loop through all devices (rack, photodiode, optical switch, …)
    for each device_index in [0 .. NB_DEVICES-1] {
```

```
        // 4. Select corresponding section name
        if device_index == 0 → section = "Rack_pilotage";
        if device_index == 1 → section = "Photodiode";
        if device_index == 2 → section = "Switch_optique";

        // 5. Read IP and Port from section
        ip = Ini_Read_String(ini, section, "TCP_address");
        port = Ini_Read_Int(ini, section, "TCP_port");

        // 6. Store values in AxisParameter array
        AxisParameter[device_index].ip_address = ip;
        AxisParameter[device_index].tcp_port   = port;
    }

    // 7. Close file and return success
    Ini_Close(ini);
    return 0;
}

//*-----------------------------------------------------------------------------------
//* Function Name : Connect_All_Devices
//* Object : use stored IP/Port to establish TCP connection with each instrument
//*-----------------------------------------------------------------------------------
void Connect_All_Devices()
{
    for each device in AxisParameter {
        ClientTCPConnect(device.ip_address, device.tcp_port);
    }
}
```

B.2. Pseudo-code: Initialization of the axes

```
//*---------------------------------------------------------------------------
//* Project : Spectral Bench Modernization
//*---------------------------------------------------------------------------
//* File Name : motion_init.c
//* Object : pseudo-code for Init_Rack() : multi-axis homing sequence
//* Creation : Intern_Mechatronics : Enzo MINACCIATTI Aug/2025
//*---------------------------------------------------------------------------
// NOTE: Full C code includes error macros, buffers, and parsing; here we keep the core
logic only.

// Conceptual data (provided elsewhere):
// - axe_params[i]: { device_id, axis_id, init_speed, status, limit_max }
// - NB_AXES_USED, NB_AXES_GOTO_SENSOR
// - Helpers: Send_zaber_command(), Decode_zaber_rep_cmd(), Add_Message_Log()

int Init_Rack(char* pc_err)
{
    // Local flags per axis
    homed_flags[NB_AXES_USED]  = {0};   // 0 = not homed, 1 = homed
    status_flags[NB_AXES_USED] = {1};   // 1 = BUSY,      0 = IDLE
    all_homed = 0;

    // (1) Disable closed loop for all axes (safer homing moves)
    for each axis i in 0..NB_AXES_USED-1 {
        device = axe_params[i].device_id;
        axis   = axe_params[i].axis_id;
        cmd = "set cloop.enable 0" on (device, axis);
        Send_zaber_command(cmd, rsp, err);
    }
    Add_Message_Log(INFO, "all cloop off");

    // (2) Drive axes toward positive end (until sensor), except those without sensor
    for each axis i in 0..NB_AXES_GOTO_SENSOR-1 {
        device = axe_params[i].device_id;
        axis   = axe_params[i].axis_id;
        v      = axe_params[i].init_speed;     // µsteps/s
        cmd = "set maxspeed v" on (device, axis);
        Send_zaber_command(cmd, rsp, err);
        cmd = "move vel +v" on (device, axis);  // constant +velocity
        // (in code: ZABER_SET_VEL used to set speed then move)
        Send_zaber_command(cmd, rsp, err);
    }
    Add_Message_Log(INFO, "all move vel sent");
```

```
    // (3) Parallel polling loop: wait until every axis is homed
    while (all_homed == 0) {
        all_homed = 1; // assume finished; will clear if any axis not done

        for each axis i in 0..NB_AXES_USED-1 {

            if (homed_flags[i] == 1) continue;  // already homed

            device = axe_params[i].device_id;
            axis   = axe_params[i].axis_id;

            // Query motion state
            cmd = "get status" on (device, axis);
            Send_zaber_command(cmd, rsp, err);
            Decode_zaber_rep_cmd(rsp);          // updates axe_params[i].status

            status_flags[i] = axe_params[i].status; // 0=IDLE, 1=BUSY

            if (status_flags[i] == 0) {
                // Axis is IDLE at positive end → issue HOME (negative direction)
                cmd = "home" on (device, axis);
                Send_zaber_command(cmd, rsp, err);

                // Restore nominal software limit (max)
                cmd = "set limit.max axe_params[i].limit_max" on (device, axis);
                Send_zaber_command(cmd, rsp, err);

                homed_flags[i] = 1;
                log "Axis i+1 : end reached -> Homing start";
            } else {
                all_homed = 0; // still moving; keep polling
            }
        }

        Sleep(1000 ms); // pause between polls
    }

    // End: all axes have started/finished their homing sequence and limits restored
    return 0; // OK
}
```

B.3. Pseudo-code: Measurement Sequence

```
//*---------------------------------------------------------------------------------
//* Function Name : Measurement_Sequence in the main function
//* Object        : Perform step-by-step motion and trigger measurements
//*---------------------------------------------------------------------------------

int main()
{
    [...]

    // Get initial positions of all active axes
    for each axis in NB_AXES_USED
        Get_Current_Position(axis);

    // Initialize iteration counter and start timer
    iteration = 0;
    start_timer();

    // Loop over positions: from 0 mm to 40 mm, step = 0.05 mm
    for position = 0.0 to 40.0 step 0.05
    {
        // Move X axis to target position
        Move_Absolute(AXE_X, position);

        // Wait until axis has stopped
        Wait_Axis_Idle(AXE_X);

        // Read encoder position and log result
        encoder_pos = Get_Current_Position(AXE_X);
        Log("Measurement %d : Encoder Position = %f mm", iteration, encoder_pos);

        // Trigger spectrum (OSA)acquisition
        Acquire_From_Analyzer();

        iteration++;
    }

    // Stop timer and log total duration
    duration = stop_timer();
    Log("Full scan duration : %f seconds", duration);
}
```

Health and Safety at Work

All experimental activities were conducted within an ISO 6 cleanroom environment, imposing strict safety, cleanliness, and electrostatic discharge (ESD) control protocols. Access to the facility required the systematic use of personal protective equipment (PPE), including cleanroom gowns, hair covers, antistatic gloves, and dedicated footwear, to prevent particulate and electrostatic contamination of optical and electronic assemblies.

During the electrical integration and testing of the prototype rack, potential risks related to stepper motor power supplies and RS-422 differential signal lines were identified and mitigated. All wiring operations were performed with power disabled, followed by continuity and insulation checks prior to re-energizing the system. All tools and instruments were used in compliance with the relevant technical datasheets and Sodern’s internal safety guidelines.

Functional validation involving motorized motion was conducted under constant supervision. No manual interaction with moving parts was permitted while the system was powered. All tests were performed with protective covers in place and emergency stop protocols defined and accessible to operators.

Sustainability and Environmental Responsibility

The developed architecture follows a reuse-oriented and sustainability-driven engineering approach. By adopting an industrial-standard motion controller and designing adapter interfaces, existing motors, encoders, and actuators were retained, avoiding unnecessary hardware replacement and material waste.

The software framework was implemented as a modular and configurable platform, allowing multiple test benches to share the same executable through an external configuration file. This minimizes redundant software development, reduces maintenance effort, and contributes to the long-term sustainability of Sodern’s optical test systems.

Furthermore, the capability to perform remote diagnostics and parameter monitoring over TCP/IP significantly decreases the need for on-site interventions and equipment transport, indirectly lowering the carbon footprint associated with maintenance and troubleshooting operations.