Automated Lens Metrology

***Abstract*—The Automated Lens Metrology system uses a fleet of sensors to measure lens properties such as diameter, center thickness, front radius of curvature, back radius of curvature, effective focal length (at different wavelengths), reflective spectra, and transmissive spectra. These properties are used to derive additional parameters and build a datasheet that provides the user with information about the inputted lens.**

***Index Terms*—Optical metrology, Documentation, Automatic testing, Spectral analysis**

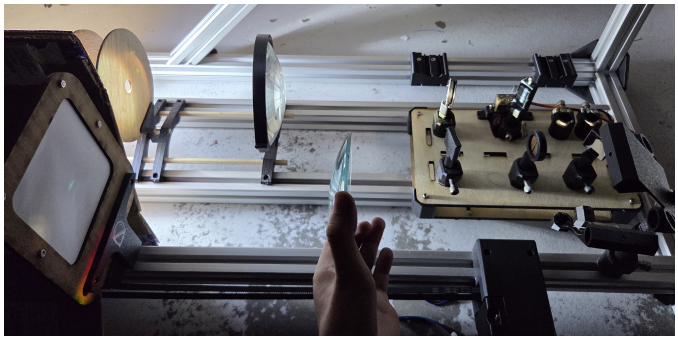


Fig. 1: Overview of optical systems.

I. Introduction

A. Motivation

Optical laboratories suffer from a chronic component organization issue. Specifically, it is hard to label and keep track of lenses that perform vastly different tasks because they tend to look very similar. As a result, there are tens of thousands of dollars of unsorted optics in our department that are not utilized effectively.

As optical engineers, we have experienced this problem first-hand. A typical sight in our labs is a teaching assistant standing over a set of lenses to compare their focal lengths by eye alone. This is an imprecise and time-consuming sorting method that is not suitable for research. Our design project aims to remedy this issue by autonomously creating a comprehensive datasheet for a given test lens. This design's intended users are students, researchers, and professors in a lab setting.

B. System Overview

This project is designed around efficient metrology. Almost all lens characteristics can be solved from the combination of geometric values, focal length, and wavelength dependency. The white light source provides a collimated beam of broadband light that can be narrowed to a single wavelength by the monochromator. Photodiodes in front and behind the test lens capture the transmissive and reflective spectra, respectively. The light passing through the test lens either converges or diverges, forming a spot on a moving panel. This spot is measured by the IAS and derives the focal length.

II. Optical Design

A. White Light Source (WLS)

The white light source (WLS) is a broadband source that provides illumination for the rest of the system. Monochromators use diffraction gratings to spatially split the light into their constituent wavelengths. In order for the interference effects to take place efficiently the light must be collimated onto the grating. Collimation is when all rays are parallel (or near to). To create the highest quality beam, the two main factors considered are source type and collimation strategy.

The source type has three main options: LEDs, incandescents, and fluorescents. LEDs emit the amalgamation of three peaks, and are popular for their high power and cheap price. Incandescents output the blackbody radiation curve, meaning output is heavily biased towards the infrared region. Finally, there are fluorescents which excite a gas to create light, and the spectrum is then based on the gas used. Most fluorescent lights people are familiar with use mercury vapor, though high-end light sources typically use xenon gas. Xenon gas discharge tubes are usually prohibitively expensive, but we realized that car headlights use the same gas, have the same spectra, and cost a fraction of standard xenon bulbs. We chose this bulb for its high intensity and flat spectra, making it the perfect source for the build.

The enemy of beam quality and collimation is etendue. Etendue is the conservation of entropy for light and is simply defined as the product of the beam area A and beam divergence Ω (see equation 1).

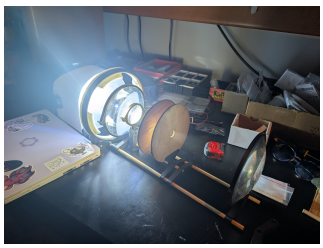
$$A\Omega \leq A'\Omega' \implies \text{complexity must increase} \quad (1)$$

To achieve a high beam quality, a sacrifice must be made: increase the beam size, decrease the divergence, increase the collimation, decrease the intensity. But before that, the light needs to get out of the bulb and into a beam expander. To do so, we chose a parabolic reflector custom-made using high temperature filament and aluminum tape. This reflector is shaped such that it maximizes the collection angle.

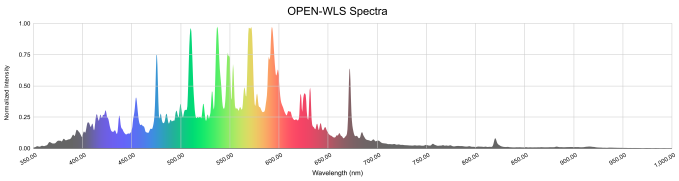
The parabolic reflector only captures and projects half the light, so to fix this we place a aspheric lens directly in front of the bulb. The asphere collimates all of the forward light into a matching collimated beam. This beam then travels forward into a set of lenses and pinholes which spatially filter the light and increase the beam size, thus decreasing the divergence. Using this method we were able to maximize the output power while achieving less than one degree divergence, and a collimated beam power of 3mW/mm^2.

(a) WLS device testing.

(b) WLS final beam.



(c) WLS final closeup.



(d) WLS spectral distribution.

Fig. 2: White light source testing.

B. Monochromator

The monochromator selects specific wavelengths from the WLS to interact with the test lens and generate data. By measuring the angles at which light is refracted as a function of wavelength, we are able to calculate the Abbe number of the test lens. By measuring the intensity of the light transmitted and reflected by the lens as a function of wavelength, we are able to estimate the lens' material.

A monochromator consists of a light source, a diffraction grating, and wavelength selection. The light source is provided by the WLS, and wavelength selection is done by mirrors and slit apertures.

The core component of a monochromator is the diffraction grating. A diffraction grating is a surface with a periodic groove structure that causes different wavelengths of light to propagate at different angles. These angles are derived from equation 2:

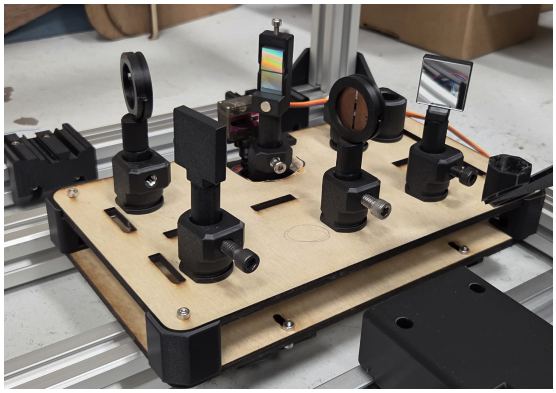
$$m\lambda = d(\sin\alpha + \sin\beta) \quad (2)$$

Where β is the diffraction angle relative to the grating's normal vector, α is the incident angle of light, m is the diffraction order, or mode, λ is the diffracted wavelength, and d is the the distance between grooves. Since the distance between grooves is inversely related to the total diffraction angle, a higher period will result in a broader spectrum. A broader spectrum means that the same exit slit will produce an output beam with a narrower wavelength selection. This led to our choice of a 1200 lines/mm grating for a spectral angular width of $16.00°$, which resulted in a measured spectral resolution of approximately 3 nm.

In order achieve different wavelengths output by the monochromator, the diffraction grating rotates to select the incident angle of white light relative to its normal. This causes the spectrum to "scan" across the static mirrors and slit apertures.

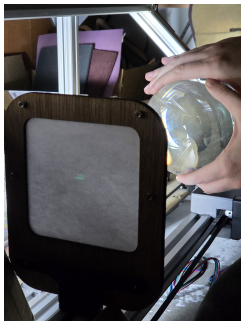
The rotation of the grating is controlled by a MG90S micro servo motor with a 5:1 gear ratio. The MG90S has a range of $180°$ and a step size of $0.5°$. With the gears, this becomes a range of $36°$ and a step size of $0.1°$ for a diffracted wavelength

step size as low as 0.8 nm. This provides excellent control for the focal length measurements at the industry standard wavelengths, and the step is well under the 5 nm step size we use for the transmissive and reflective spectra.



(a) Monochromator layout.

(b) Diffraction grating gear mount.



(c) Monochromator output with test lens.

Fig. 3: Monochromator testing.

C. Image Acquisition System (IAS)

The Image Acquisition System (IAS) takes a photo of the image plane at multiple distances. From those images spot sizes are extracted using opencv circle fitting to the brightest spot in the image. By modeling the change in spot size over distance the focal length is extracted for a given wavelength.

The IAS uses a HQ Raspberry Pi Camera sensor, which is a 12.3 megapixel CMOS chip. The light is taken in through a pinhole, focused by a lens, and projected onto the sensor.

We chose a basic pinhole and lens combo, as that provides the highest field of view with relatively low distortion. The pinhole is 1mm in diameter, and the focusing lens has a EFL of 10mm. This compresses a 100x100mm plane down to the 7.857mm sensor. See fig. 4.

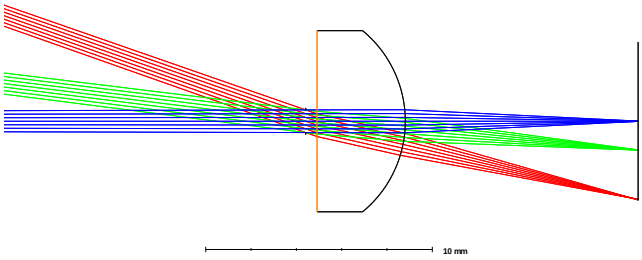


Fig. 4: Zemax ray simulation of IAS.

There are two types of spherical distortion barrel distortion which is where it seems the center of the image is moving towards the viewer and is caused by the pinhole being placed in front of the lens, and pincushion which is the reverse.

For our system we chose to use a forward pinhole and thus have a maximum distortion of -3% on the far edges of the system. Since the spot is in the center of the image the affect is slight, however can be compensated in software on the image processing side.

In order to allow easy calibration, focusing, and element placement the lens and pinhole were placed in a screw-like mechanism to allow distance adjustment. See fig. 5.

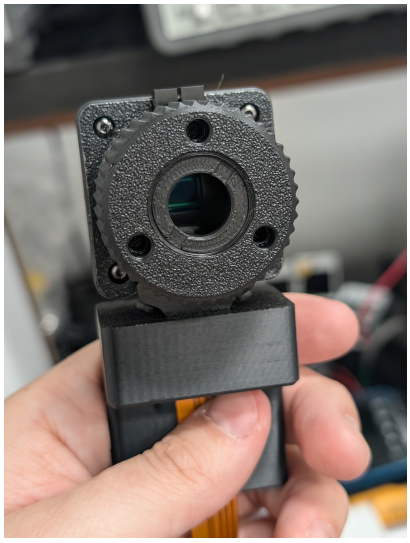


Fig. 5: Photo of IAS assembly.

D. Equations & Datasheet Processing

The raspberry pi outputs to a series of STM8s and set values to take spot size measurements. Once those spot sizes, geometric values, and spectral values are attained, the raw data is exported to a csv file. This raw csv enters a processing function which derives a bunch of values for example IOR based off the raw data. This processed data is output as a processed csv. A map of the calculations is visible in fig. 6.

Finally we are ready to generate the datasheet. A python file takes in a series of LaTeX documents which reference the processed csv to generate subfigures, for example the spectra, or geometry. Finally those figures are encorporated into a larger .tex document. This modular nature means editing one figure doesn't break the entire datasheet. Finally the .tex file is compiled via PyLaTeX and exported as a pdf to be displayed. See a sample datasheet in fig. 7.

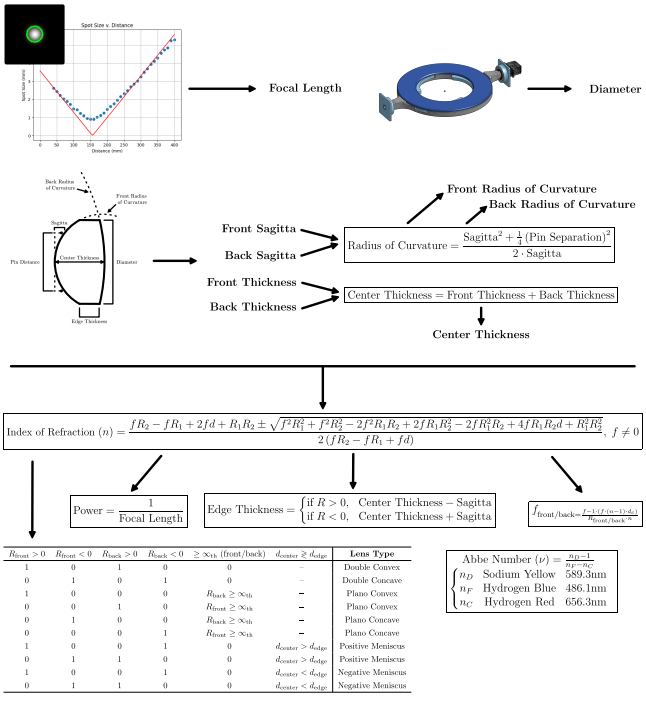


Fig. 6: Optical calculation relationship map.

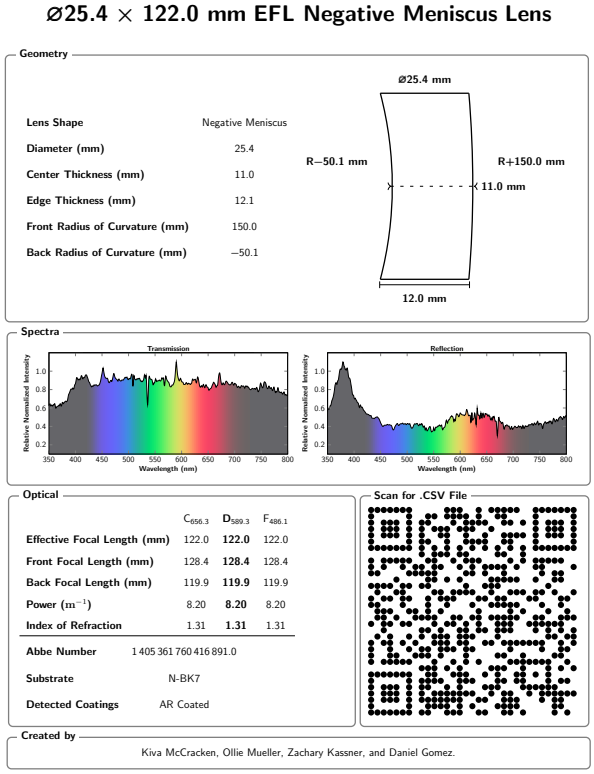


Fig. 7: Generated datasheet.

III. Electrical Systems

A. Processor Selection

1) Raspberry Pi 5

The Raspberry Pi 5 was selected as the central system controller after evaluating it alongside the Pi 3, Pi 4, and NVIDIA Jetson Nano [1]. Its 2.4 GHz quad-core ARM Cortex-A76 processor and LPDDR4X memory provide substantially higher single-thread performance than earlier Pi models, which is important for the concurrent demands of I^{2}C bus management [2], image acquisition [3], photodiode sampling, encoder reading, and real-time datasheet generation. The Pi 5 also exposes dual full-speed MIPI CSI connectors and a PCIe 2.0 interface, giving headroom for future camera or peripheral upgrades. Compared with the Jetson Nano, which is optimised for GPU-accelerated AI inference, the Pi 5 offers a better balance of general-purpose compute, power draw, and cost for the orchestration-heavy workload of this system. The

Pi 5 runs standard Raspberry Pi OS and is natively compatible with the Python libraries already chosen for the project: *smbus2* [4], *picamera2* [3], *pandas* [5], and *opencv-python* [6].

2) STM8 Microcontrollers

Each hardware task requiring deterministic, real-time control is handled by a dedicated STM8S microcontroller [7]. The STM8 family was chosen over the STM32 for this role because its lower cost and simpler peripheral set are well-matched to single-purpose slave nodes that do nothing beyond driving a motor or reading a sensor. Tasks that require only a handful of GPIOs, one timer, and an I^2C peripheral do not justify a more capable processor. The STM8's Standard Peripheral Library provides sufficient hardware abstraction for the firmware needed here, and the Cosmic CXSTM8 toolchain integrates directly with ST Visual Develop for straightforward programming and debugging.

B. Electrical Sub-System Objectives

- **Raspberry Pi 5** — Central coordinator, managing communication across all electrical subsystems.
- **Stepper Motor 1** — Drives the imaging carriage along the x-axis of the linear gantry.
- **Stepper Motor 2** — Controls rotational positioning of the measured lens about the z-axis.
- **Micro Servo** — Governs diffraction grating rotation within the monochromator.
- **Monitor, Buttons, & Buzzer** — Provide user interaction and system feedback.

C. Power

The ALM system draws power across several voltage domains to support its mix of digital, analog, and electromechanical subsystems. The supply voltage, current draw, and power consumption of each subsystem is sumarized below.

The system connects to the power strip via two cables: one supplying the white light source and one supplying all remaining DC-powered components. Separating the AC and DC connections during early prototyping keeps the high-voltage mains circuitry physically and electrically isolated from the low-voltage digital and motor drive electronics, reducing the risk of ground loops, noise coupling, and potentially dangerous fault conditions while debugging. This separation also makes it easier to safely disconnect and probe individual subsystems without inadvertently exposing sensitive components to line voltage. The power for the rest of the DC system is in accordance with the block diagram presented in figure 7.

D. Stepper Motor Subsystem

The stepper motor controller subsystem represents one of the more involved hardware blocks developed by the group, integrating a switching regulator, STM8, and A4988 stepper driver IC into individual modules. These modules are interconnected via 2.54 mm female headers on a motherboard, allowing individual boards to be removed and replaced for prototyping and repair.

The stepper motor controller block receives 12 V from the main system supply, which is stepped down to a regulated 5 V rail using a TPS562201 synchronous buck regulator. The 5 V rail it produces serves as the primary power input for the downstream circuitry on the board.

The STM8 microcontroller serves as a local processor for the system. It controls the STEP, DIR, and ENABLE signals that command the motor driver. It is supported by its own onboard 3.3 V linear regulator derived from the 5 V rail. Communication with the broader system is handled over UART to the Raspberry Pi.

The A4988 is the stepper motor driver IC that translates the STM8's digital control signals into the high-current in the stepper motor. It also carries its own 3.3 V regulation for its internal logic, while the motor draws directly from the 12 V rail to deliver the higher power needed. The A4988 handles current regulation, microstepping, and protection internally, making it a self-contained drive stage that requires only simple digital inputs from the microcontroller to operate.

IV. Integration & Automation

A. System Architecture

The ALM system uses a distributed control architecture: the Raspberry Pi 5 acts as the I^2C master and high-level sequencer, while up to four STM8 slave nodes handle dedicated hardware tasks [2]. This keeps time-critical motor and sensor loops on

TABLE I: Electrical System Power Budget

Component	#	Manufacturer	Coupling	V	mA	mW
Stepper Motor Systems	2	ALM Group	DC	12	550	6600
Micro Servo	1	Beffkkip	DC	5	200	1000
Raspberry Pi	1	Raspberry Pi Ltd	DC	5	1500	7500
Raspberry Pi Camera	1	Sony	DC	5	250	1250
Monitor	1	DHPL	DC	5	1500	7500
Buzzer	1	ALM Group	DC	12	25	300
Photodiode PCB	1	Thorlabs	DC	12	50	600
White Light Source	1	ALM Group	AC	120_{RMS}	330_{RMS}	40000

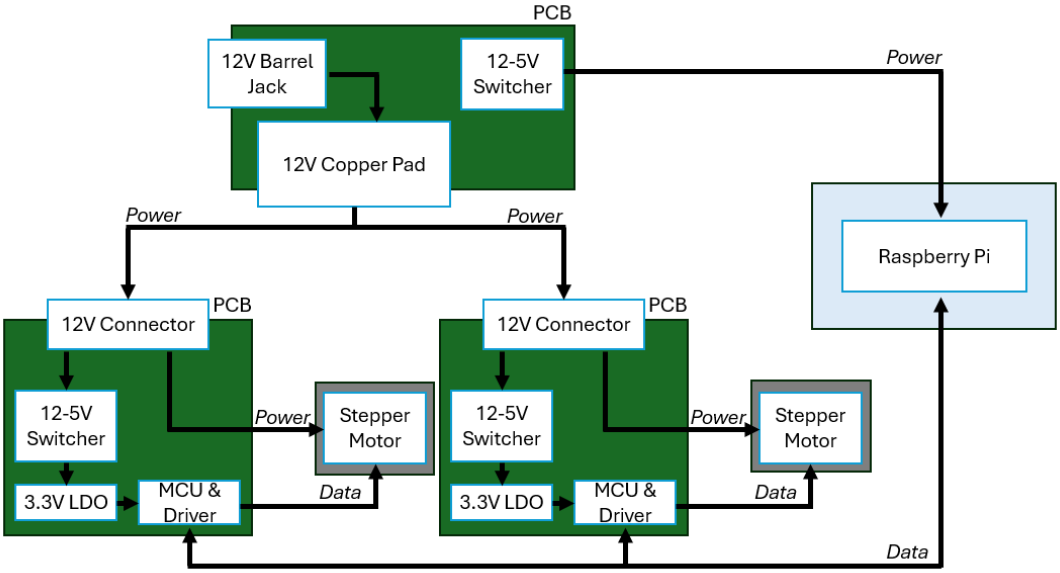


Fig. 8: DC System Power Block Diagram

bare-metal firmware while leaving Python on the Pi to manage sequencing, data aggregation, and output generation. The I^2C bus was chosen over SPI and UART for this multi-node topology because it requires only two shared lines (SDA/SCL) regardless of how many slaves are present, with each device uniquely identified by a 7-bit address — no additional chip-select or address field inside the packet is needed. The four planned slave nodes are listed in Table II.

TABLE II: Planned I^2C Bus Device Map

Address	Role
`0x10`	Gantry / stepper motor (imaging carriage)
`0x20`	Monochromator servo control
`0x30`	Photosensor data acquisition
`0x40`	Lens arm stepper (gripper positioning)

New slave nodes can be added to the bus without modifying any existing firmware or packet format; only a unique address must be assigned.

B. Communication Protocol

All inter-device communication follows a lightweight custom framing scheme. Every packet, in either direction, obeys the fixed layout:

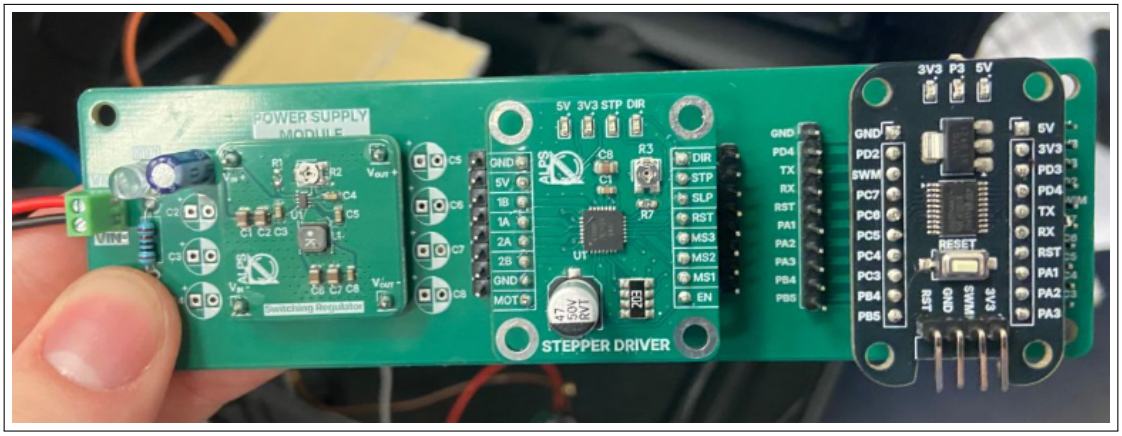


Fig. 9: Stepper Motor Controller Motherboard

$$\underbrace{\texttt{LEN}}_{1\,\text{B}}\ \underbrace{\texttt{TYPE}}_{1\,\text{B}}\ \underbrace{\texttt{HCKSUM}}_{1\,\text{B}}\ \underbrace{\texttt{DATA}}_{\texttt{LEN}\,\text{B}}\ \underbrace{\texttt{DCKSUM}}_{1\,\text{B}} \tag{3}$$

where $\texttt{HCKSUM} = \texttt{LEN} \oplus \texttt{TYPE}$ and DCKSUM is the XOR of all payload bytes. Both checksum fields are omitted when $\texttt{LEN} = 0$. The receiver discards any packet whose header checksum fails, providing a first line of defence against bus noise or framing errors. Each acknowledgement from a slave echoes the TYPE byte of the command being confirmed, allowing the Pi to verify receipt even when multiple slaves respond independently.

TYPE codes are organised by functional domain. The 0x0x range covers control and acknowledgement messages common to all nodes. The 0x1x range carries asynchronous notifications such as motion completion events. The 0x2x–0x3x range is reserved for stepper motor commands (absolute position, homing, status queries). The 0x4x range covers servo angle commands for the monochromator. Additional ranges can be allocated for photosensor queries and arm control as those nodes are brought up.

C. Gantry Controller

The gantry subsystem translates the imaging carriage along approximately 400 mm of aluminium extrusion, driven by a NEMA 23 stepper motor through a belt-and-pulley stage. The STM8 firmware [7] implements a fully polling-based I^{2}C slave: the main loop checks the status register on every iteration and services the bus state machine without using interrupts. This approach was adopted after interrupt-driven variants consistently failed to appear on i2cdetect; switching to polling resolved address detection immediately and has remained stable.

Position commands arrive as a 16-bit fixed-point value (mm$\times$10, big-endian). A non-blocking state machine handles homing, endstop backoff, and point-to-point moves, ensuring that the I^{2}C bus is always serviced between motion ticks. The system homes automatically on first use: on power-up the carriage drives toward a limit switch, backs off by a fixed number of steps once contact is made, and sets that position as the origin. All subsequent moves are issued as absolute coordinates.

1) Gantry Test Results

The gantry was characterised by commanding three target distances (50 mm, 215 mm, and 400 mm) across repeated runs while varying the steps-per-mm calibration constant. Three values were evaluated: 55, 52, and 58 steps/mm. Mean measured position, standard deviation, and absolute error were computed for each combination and are summarised in Table III. The 55 steps/mm setting produced the best overall accuracy, achieving sub-millimetre error at the 50 mm and 215 mm targets and a worst-case error of -3.75 mm ($< 1\%$) at 400 mm. The current prototype lacks lateral stabilisation in the PTFE sliders, which contributed to the increased deviation at longer travel; this will be addressed in the final build. The 10 mm imaging step requirement is comfortably met by the calibrated configuration.

TABLE III: Gantry Positioning Summary (mean of repeated runs)

Steps/mm	Target (mm)	Mean (mm)	σ (mm)	Error (mm)	Error (%)
3*55	50	50.0	0.00	0.00	0.00
	215	214.5	0.58	−0.50	−0.23
	400	396.3	1.26	−3.75	−0.94
3*52	50	51.0	0.00	+1.00	+2.00
	215	202.0	0.00	−13.0	−6.05
	400	375.0	0.00	−25.0	−6.25
3*58	50	52.0	0.00	+2.00	+4.00
	215	227.0	0.00	+12.0	+5.58
	400	420.0	0.00	+20.0	+5.00

D. Monochromator Servo Controller

The monochromator's diffraction grating is rotated by a servo motor under I²C command [7], allowing the Pi to sweep the output wavelength across the visible spectrum during a measurement run. Angles are transmitted as integer tenths of a degree to avoid floating-point arithmetic on both ends, and the STM8 converts the received value into the appropriate PWM duty cycle using integer arithmetic. The same generalised framing protocol is used for servo commands as for all other nodes, with a dedicated TYPE code in the `0x4x` range.

E. Raspberry Pi Software Responsibilities

The Pi 5 is responsible for every layer of the system above raw hardware actuation.

I²C bus management. All four STM8 nodes are addressed over a single shared bus via the *smbus2* library [4] using raw `i2c_rdwr()` transactions. SMBus-layer helper calls were found to prepend an extra register byte that corrupts the frame parser on the STM8 side and were abandoned.

Encoder reading. Lens diameter is measured by a KY-040 rotary encoder module embedded in the self-centering gripper arm assembly. The encoder connects directly to the Pi's GPIO pins [1]. As the gripper closes around a lens, the Pi counts pulses from the encoder to calculate the arm displacement, from which the lens diameter is derived.

Photosensor acquisition. Transmissive and reflective intensity readings are collected from the photosensor node at each measurement wavelength. The Pi issues a sensor-request command, reads the response, and stores the intensity value alongside the corresponding wavelength before advancing to the next grating angle.

Measurement sequencing. The Pi orchestrates the full measurement loop [2]: commanding the gantry to each 10 mm step position, sweeping the monochromator through the target wavelengths at each position, and collecting IAS images [3] and photodiode readings at each combination. Motion at one node is always confirmed with an acknowledgement or completion notification before the next command is issued. Concurrent tasks — serial communication, data logging, and image acquisition — are managed using Python's *asyncio* library [8], with structured runtime logging provided by the *logging* module [9].

Datasheet generation. Once all raw data are collected, a Python pipeline processes images using *opencv-python* [6] and *Pillow* [10] to extract spot size, aggregates geometric and spectral measurements with *pandas* [5], derives secondary parameters (effective focal length, index of refraction, Abbe number), and exports the results to CSV [11] or XLSX [12] format. A LaTeX rendering pass then produces the final human-readable datasheet.

User interface. The Pi 5 drives a connected monitor that displays system status, measurement progress, and any error conditions in real time. Physical buttons connected to the Pi's GPIO trigger key stages of the sequence, including initiating a measurement run and confirming lens placement.

This is so that a complete measurement session requires no keyboard or mouse interaction from the operator.

REFERENCES

[1] Raspberry Pi Foundation, *Raspberry Pi Documentation*, 2024, accessed: 2025-10-23. [Online]. Available: https://www.raspberrypi.com/documentation/

[2] R. P. Foundation, *Raspberry Pi I^2C Interface Documentation*, 2024.

[3] ——, "Picamera2 python library," https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf, 2024, accessed: 2025-09-05.

[4] L. Cipriani, "smbus2: A drop-in replacement for smbus-cffi / smbus-python," https://smbus2.readthedocs.io/, 2024, accessed: 2025-09-05.

[5] N. Inc., "Python data analysis library," https://pandas.pydata.org/, 2024, accessed: 2025-09-05.

[6] O. team, "Opencv-python: Open source computer vision library," https://opencv.org/, 2024, accessed: 2025-09-05.

[7] STMicroelectronics, *STM8S003F3/K3 Value Line, 16-MHz STM8S 8-Bit MCU Datasheet*, 2018, dS7147 Rev 10. [Online]. Available: https://www.st.com/resource/en/datasheet/stm8s003f3.pdf

[8] P. S. Foundation, "asyncio — asynchronous i/o," https://python-pillow.org/, 2024, accessed: 2025-09-05.

[9] ——, "logging — logging facility for python," https://docs.python.org/3/library/logging.html, 2024, accessed: 2025-09-05.

[10] A. Clark and contributors, "Pillow: Python imaging library," https://pillow.readthedocs.io/en/stable/, 2024, accessed: 2025-09-05.

[11] P. S. Foundation, "Csv file reading and writing," https://docs.python.org/3/library/csv.html, 2025.

[12] E. Gazoni and contributors, "openpyxl: A python library to read/write excel 2010 xlsx/xlsm files," https://openpyxl.readthedocs.io/, 2024, accessed: 2025-09-05.