İSTANBUL UNIVERSITY PRESS

Physics and Astronomy Reports
PAR 2025, 3 (2), 94–99
DOI: 10.26650/PAR.2025.00010

RESEARCH ARTICLE

Automation of Measurements of A Stepper Motor-driven Optical Monochromator Using Python

Ş. Şeker [1,2], P. Schwarz [2], C. Weidner [2], and S. Kröger [2*]

[1]Middle East Technical University, Faculty of Art and Sciences, Department of Physics, Ankara, Türkiye
[2]Hochschule für Technik und Wirtschaft Berlin, Fachbereich 1, Wilhelminenhofstr. 75A, D-12459 Berlin, Germany

ABSTRACT
This project presents automatic control of measurements of a stepper motor-driven classic optical grating monochromator that is more than 50 years old. The automation of measurements was realized by the use of a stepper motor and the corresponding Python programming. The Python code, which follows the general outline of an earlier LabVIEW design, also includes a graphical user interface and an additional app for evaluation. The Python software is built in separate parts that work together. A central "state manager" (Finite State Machine, FSM) organizes and follows each step of the process. Other background parts handle specific tasks, such as moving the motor of the monochromator or collecting measurement data, while the main program keeps the user interface active and responsive. Besides allowing the user to manually change the wavelength step by step, the program can also perform automatic scans or carry out several scans in sequence from a list created by the user. This work enables the integration of high-quality historical optical instruments with modern experimental setups – for example, in high-resolution laser-induced fluorescence spectroscopy. The project is explained here, and the program code is shared publicly on GitHub as part of open science (Şeker et al. 2025). Test measurements using various lamps are presented.

Keywords: Spectroscopy, monochromator, python, GUI

1. INTRODUCTION

There is a great need for experimental data on the structure of complex atoms (fine structure, hyperfine structure, and isotope shift), such as those collected, for example, by the NIST Atomic Spectra Database and made available to science (Kramida et al. 2025). Complex atoms refer to atoms with multiple open valence electron shells in excited states, especially transition metals from the transition groups. Data on the structure of complex atoms are not only of great interest for applications in astrophysics, but also provide further insight into effects in the area of fundamental physics, such as second-order effects in quantum physics. The quantity and accuracy of the experimental data play a key role in future optimization of the theoretical description, both in ab initio calculations and in semi-empirical analyses of atomic structures. Until now, high-resolution laser spectroscopic measurements have been a very time-consuming, computer-assisted, manual process. Over the past decade, there have been significant advances in the development of narrow-band, tunable laser light sources that open up new possibilities for automated measurements in laser spectroscopy. For some applications, such as laser-induced fluorescence spectroscopy, in addition to a laser, another optical device is required for the spectral decomposition of the fluorescent light. Optical grating monochromators are suitable for this purpose. Even if such devices are older, they are, despite their age, still excellent optical instruments of timeless quality and high precision, which can be used in a wide variety of applications. The only problem for use in automated setups: Older devices are generally motorized, but can only be operated manually. Therefore, they are not suitable for computer-controlled applications.

The software presented here demonstrates how older spectrometers can be upgraded using modern Python control and a stepper motor, without changing their optical parts. The software uses a lightweight structure based on Python tools: Tkinter provides an easy-to-use graphical interface, Matplotlib displays live measurement plots, NI-DAQmx ensures reliable data collection, and pySerial handles the serial communication with the stepper motor controller (see Section 3). The program follows a structure similar to a previously used LabVIEW design. A simple process manager (called a Finite State Machine) organizes all operations. Background tasks take care of hardware control, while the main loop keeps the user interface active. Each measurement cycle – from setup and data collection to stopping safely – corresponds to the original LabVIEW func-

Corresponding Author: S. Kröger **E-mail:** sophie.kroeger@htw-berlin.de
Submitted: 16.11.2025 • **Revision Requested:** 08.12.2025 • **Last Revision Received:** 09.12.2025 • **Accepted:** 09.12.2025 • **Published Online:** 31.12.2025

tions. This design provides smooth control, consistent and well-documented measurement data, automatic repeated scans, and safe shutdown procedures. It makes the transition from manual to automated use straightforward.

The complete software and its documentation are shared openly to support reuse and learning by others. The development of tools such as this Python application and its dissemination through publications and open source tools helps to disseminate existing expertise.

2. EXPERIMENTAL SETUP

The essential components of the experimental setup (the monochromator, stepper motor, detectors, and analog-digital converter) are briefly described below. While the Python program is demonstrated with this specific hardware, it can be easily adapted – with minor adjustments – to other comparable setups.

Monochromator

Grating monochromators use a diffraction grating as the dispersive element. Since these instruments do not require refractive elements, they can be used over a very wide wavelength range. The optical elements (mirrors and grating) and the mechanical construction are not subject to significant wear and tear; high-quality devices do not suffer deterioration in quality due to age, as far as it concerns the optical and mechanical. The optical device used here is a Czerny-Turner grating monochromator, Model 2051 from McPherson, which is more than 50 years old. An advertisement in the journal *Physics Today* from 1971 described the device as: 'The new McPherson Model 2051 World's most versatile one-meter precision scanning monochromator and spectrograph' (Kronberger 1971). This monochromator is specified by 1 m focal length, an aperture ratio (f-number) of 8.6, an Echelle grating (110 mm × 110 mm) with 1200 g mm^{-1} (grooves per mm), resulting in an accuracy of 0.05 nm and a reproducibility of 0.005 nm. The grating covers the wavelength range from 185 nm to 1.3 µm.

Stepper motor

To enable automation, the original factory-installed AC motor was replaced by a computer-controlled stepper motor. It consists of a brushless DC motor with integrated motion control and RS232 interface from Faulhaber, which is equipped with a 29:1 gearbox. The Xon/Xoff protocol is used for fast command transmission (see Faulhaber 2014). The original adjustment gear was retained in this setting. To prevent driving over the edge, limit switches were installed.

Detectors

The monochromator offers two output ports that can be switched via a foldable mirror at the output slits. Two different detectors were attached to the two output ports. Either a photomultiplier or a photodiode was available as a detector. The model H9306-04 from Hamamatsu was used as a photomultiplier with a wavelength range 185 nm to 830 nm and a peak wavelength of 530 nm, and the silicon photo diode PDA100 from Thorlabs with a wavelength range 340 nm to 1100 nm.

A-D-converter

As an analogue-digital (A-D) converter, the module NI-USB-6009 DAQ from National Instruments is used (National Instruments 2025a). The device is a bus-powered multifunctional USB data acquisition device with a resolution of 14-bit and a sampling rate of 48 kS s^{-1}. It is equipped with eight analogue inputs, two analogue outputs, and other connections that are not relevant to our application. From eight input channels, only two are currently used.

3. PYTHON IMPLEMENTATION

3.1. System Overview

The application was built in Python 3 (Python Software Foundation 2025a), a general-purpose programming language known for being readable and available on all major operating systems with the right imports and a few adjustments. Python was mainly chosen because of it being available free of charge while quickly moving from ideas to being working tools. The Python libraries contain built-in modules that provide access to system functionality (Python Software Foundation 2025b), which we used a few in the process of writing the program:

- The standard libraries were used for configuration and data handling (e.g., Json, CSV, Pathlib) and for reliable diagnostics via logging (Python Software Foundation 2025c). This keeps the tool easy to install and portable across operating systems, with mature documentation and long-term support in need.
- The specific toolkit for GUI (Graphical User Interface) Tkinter (Python Software Foundation 2025d) was used, a built-in interface for Windows. The details of the GUI system can be seen in Section 3.3, but to mention briefly, Tkinter provides native-looking widgets and a straightforward event module, so the interface remains simple to operate and easy to maintain, besides being free.
- For plotting the standard module Matplotlib (Matplotlib 2025) was used – also embedded in tkinter; live plots and save figures are provided by Matplotlib, which can be accessed via TkAgg backend.
- In the data acquisition process, NI-DAQmx (National Instruments 2025b) was used. Instruments I/O with National Instruments hardware is handled through the official supported "nidaqmx" package. The library follows NI-DAQmx's tasks module (tasks, channels, timing, reads/writes), which is leanly mapped to laboratory workflows and NI documentations (National Instruments 2025c), ensuring predictable behavior and straightforward troubleshooting.
- The serial connection, which is one of the core parts of the whole code, was handled via pySeria (Python Software Foundation 2025e), where devices expose a serial (COM/USB-serial) interface that is used to open ports, communicate with the motor, send ASCII commands, and read responses in a cross-platform way. Its simple API and strong documentation make it a reliable choice for instruments.

3.2. Architecture

The architecture deliberately mirrors the previously existing LabVIEW application, keeping the same operator flow while translating it into Python. The design is centered on three pillars:

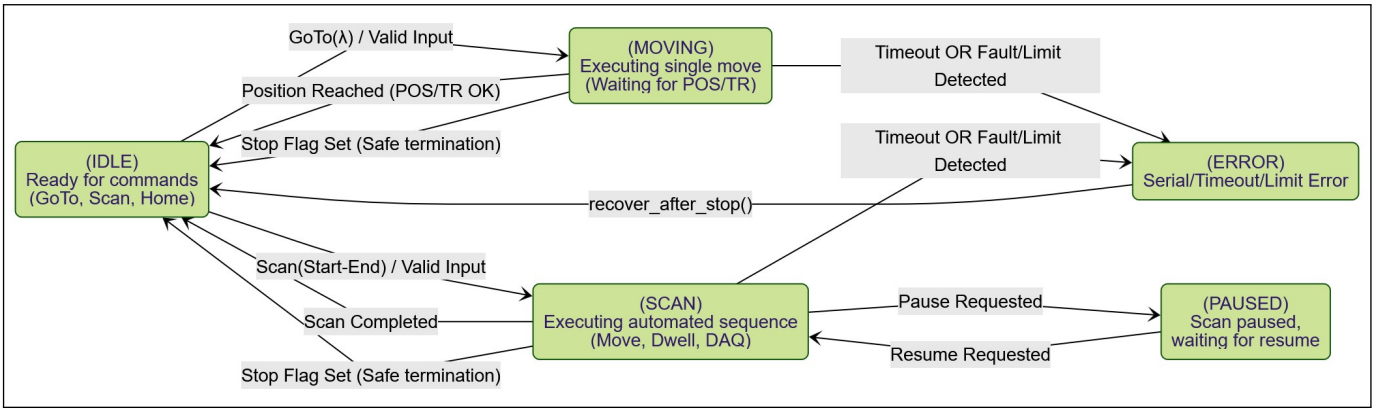

Figure 1. State chart of a finite state machine of monochromator control.

- a finite state machine (FSM) to make the behavior explicit,
- background workers to emulate LabVIEW's parallel loops, and
- layered safety so every run begins and ends in a known state.

This preserved the functionality of the original buttons, scan steps, and stop behavior while making the system portable and maintainable.

System behavior is organized into a compact set of states – IDLE, MOVING, SCAN, PAUSE, STOP and ERROR – with transitions triggered by user actions and device events. This is a direct Python analogue of LabVIEW's Queued State Machine/ Queued Message Handler: events are enqueued, a controller dequeues them, runs the associated action, and moves to the next state National Instruments (2025d). A state chart as a schematic representation of the FSM with its states and its key transitions is shown in Figure 1. Using FSM keeps intent visible, simplifies testing, and aligns the state chart formalism that underpins many LabVIEW patterns.

To keep the "front panel" responsive, long operations – motion commands, reads from the analogue-digital converter (DAQ), file saves – run in background workers while the main UI loop handles input and drawings. This mirrors LabVIEW's Producers/Consumer and multi-loop templates: producers (UI/events) push work; consumers (workers) execute, and report results back. In Python, we implement this with threading for workers and Tkinter's timed callback (after) for periodic UI updates, which achieves the same decoupling of rates that can be seen in LabVIEW's parallel loops.

Safety follows LabVIEW's layered practice. Before acquisitions, we validate device presence and settings. During acquisitions, we honor soft limits, timeouts, and immediate STOP/PAUSE, so the device is maintained safe, and operators can always recover cleanly. After acquisitions (or on faults), we follow NI-DAQmx's task lifecycle: stop and then clear tasks to release resources. This deterministic teardown prevents 'half-open' tasks and ensures the next acquisition starts from a known good state.

This process replicates the familiar logic of a LabVIEW design. A typical scan begins in the IDLE state, moves to the start point, and transitions to MOVE-ING to verify completion. During the SCAN state, the system acquires data (with dwell/averaging if set), updates the display, and advances. At any moment, PAUSE/STOP commands initiate a safe exit, returning the system to IDLE.

While the underlying technology is different, the contract with the operator and the discipline of message-driven, state-based control remain the same.

3.3. Graphical user interface

The programme presented here does not use a command interface, but rather a graphical user interface (GUI). The GUI (shown Figure 2) contains buttons for input and output of information that is exchanged with the user. Each button is wired to a Tkinter command callback that validates inputs, posts an action to the controller (FSM), and returns immediately so the window stays responsive; a *lightweight after timer* then refreshes labels, enables or disables controls as appropriate, and redraws the plot. In this scheme,

- ***Connect*** opens the device link (serial and DAQ) and performs controller initialization, while
- ***Disconnect*** runs a safe shutdown before closing the port;
- ***Go To*** commands a single move to the requested wavelength and updates the current position once motion completes;
- ***Start Scan*** executes a linear scan defined by Start, End, Step, and Wait, Pause temporarily suspends acquisition,
- ***Resume*** continues from the same point, and Stop aborts safely and returns the system to a stable idle state;
- ***Jog - and Jog +*** nudge the mechanism by the configured jog step for quick manual alignment;
- ***Mark Point*** acquires one averaged reading at the present wavelength and marks it on the plot for reference;
- ***Save CSV*** exports the current dataset immediately, while
- ***Auto-save*** performs the export automatically at the end of each scan;
- the ***Scan Queue*** controls – Add, Run, and Clear -— let the user build a list of scans to execute sequentially.

Throughout, the triggering mechanism remains the same: button press → callback → queued action → periodic UI update.

The benefits and logic behind this GUI are pragmatic: the layout mirrors the laboratory workflow (connect → position → scan → save), which reduces cognitive load and training time, while input validation and context-sensitive en-

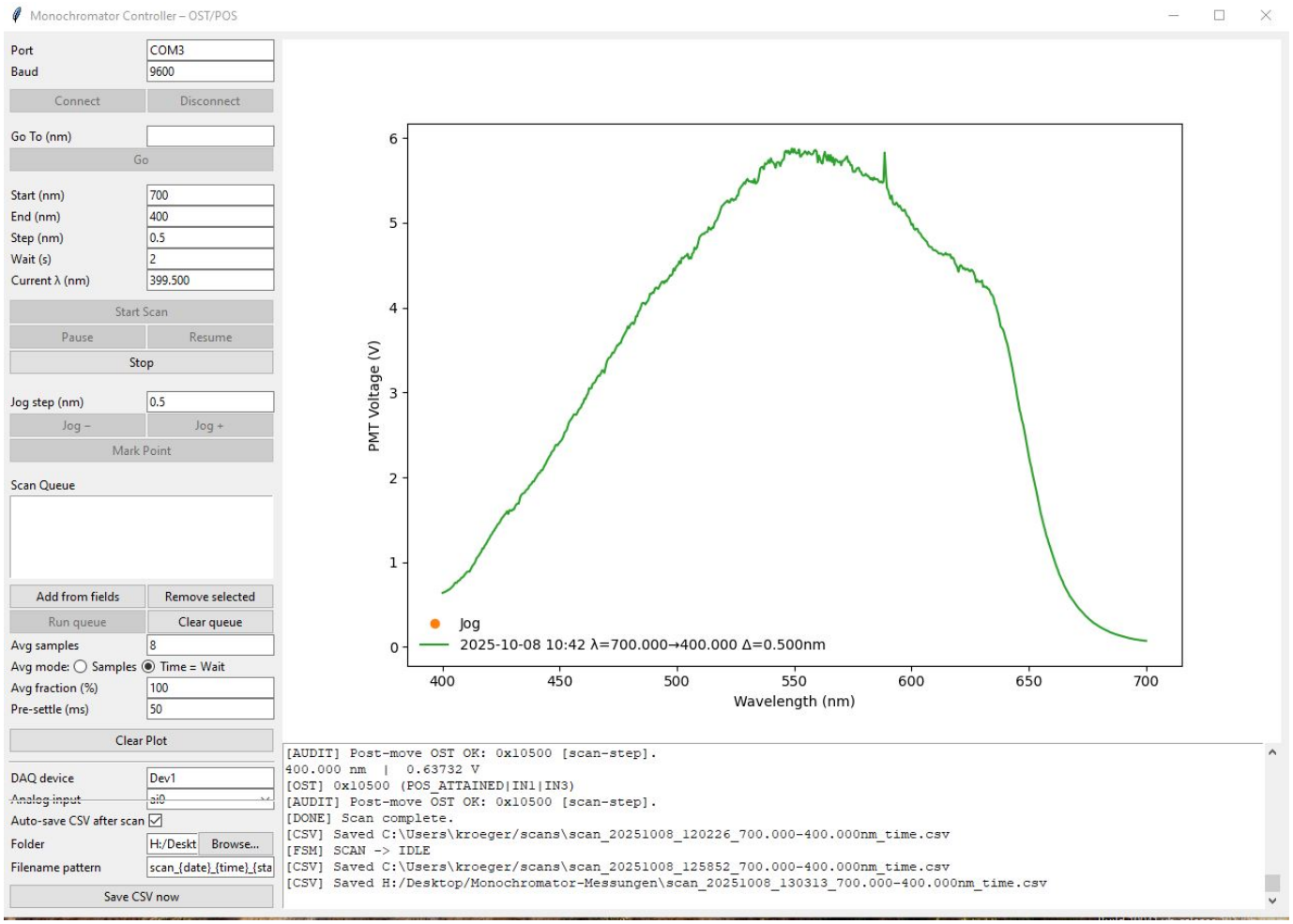

Figure 2. Screenshot of the graphical user interface (GUI).

able/disable prevent invalid actions and common operator errors. Separating momentary actions (Go To, Jog) from process controls(Start/Pause/Resume/Stop) makes states and responsibilities clear: momentary actions change position once; process controls govern long-running acquisition with immediate, safe interruption paths. Pause/Resume exists to preserve experimental context (same configuration, same point) without tearing down hardware, whereas Stop guarantees a deterministic return to a known-good state. Mark Point supports quick diagnostics without committing to a full scan; Auto-save and Save CSV enforce reproducibility and reduce data-loss risk by making persistence explicit and automatic; and the Scan Queue enables unattended, repeatable batches – improving throughput while keeping operator intent visible and auditable. Overall, the design prioritizes responsiveness, safety, and reproducibility: every button has a single, obvious purpose; every long task remains interruptible; and every run leaves a traceable record and a stable instrument.

3.4. Mechanical backlash

A gearbox connects the stepper motor to the high-precision spindle adjustment of the grating. This gearbox exhibits slight backlash (or slip). This backlash causes the motor to rotate briefly when reversing direction without the spindle adjustment being activated. This in turn leads to a wavelength offset when reversing the measurement direction (i.e., from ascending to descending wavelengths or vice versa). This wavelength shift, caused by the backlash, is highly reproducible. To compensate for this offset, the software tracks the current direction of rotation. Each time the direction is reversed, the motor is first moved several steps corresponding to the backlash.

4. APPLICATION EXAMPLES

Four spectra are shown as application examples in Figure 3. The first two Figures 3a and 3b show spectra from a low-pressure spectral lamp for the two elements helium and mercury. Here, the characteristic, sharply defined emission lines of the respective element can be seen. When the monochromator is used in combination with a tunable laser for laser-induced fluorescence measurements in the future, the spectra will look similar to these spectra. Depending on the selected slit width, this monochromator can achieve a spectral resolution of 0.02 nm in the visible wavelength range.

Of course, there are also applications for the monochromator where high resolution is not absolutely necessary. Examples of this are shown in Figures 3c and 3d, which display the spectra for two light media. Both lamps emit white light to the human eye, but have very different spectra. Figure 3c shows the spectrum of an energy-saving lamp. Its light is generated in a two-stage process where electrical energy is converted into ultraviolet (UV) light, which is then converted into visible light

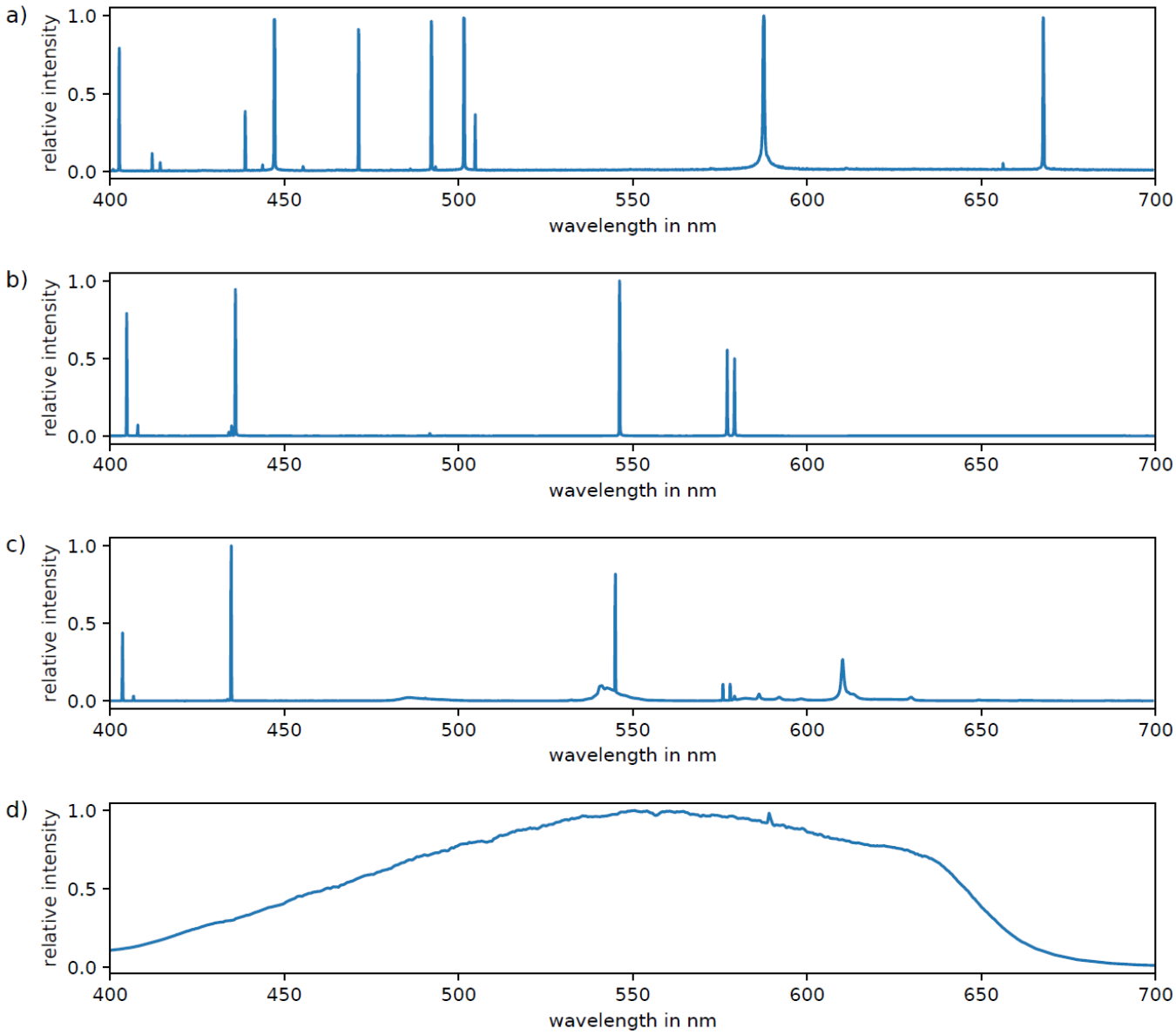

Figure 3. Example spectra recorded in the entire visible wavelength range from 400 nm to 700 nm for a) helium spectral lamp, b) mercury spectral lamp, c) an energy-saving lamp, and d) a halogen lamp.

by a fluorescent substance. This results in a spectrum that is a combination of sharp lines and broader emission ranges.

Figure 3d shows the continuous spectrum of a (high-pressure) halogen lamp. The distinct additional emission peak, which is visible in this spectrum at around 589 nm, comes from the sodium D lines, which outshine the continuous light even with the smallest amounts of sodium compounds in a lamp. The drop in intensity of the continuous spectrum from around 650 nm, which is not to be expected with a thermal light source, is caused by the reflector in the lamp. This dichroic reflector, also known as a cold light mirror, allows most of the infrared to pass through the reflector in order to prevent heating when irradiated with the light of this lamp.

5. FUTURE WORK

Future work includes several ideas to enhance both functionality and safety. Key improvements involve using the controller's statusword as the primary motion completion check, auto-tuning motion times based on real move durations, and adding real-time peak fitting during scans. To improve usability, features like a single-file installer, acquisition presets, and a background data writer for streaming results with metadata are planned. A Run Sheet export for session summaries, a scan-queue editor, and plot comparison tools will further streamline workflows. To enhance safety, adding a pre-run checklist gate for critical fields, a motion sanity guard to block out-of-bounds moves, and a unified watchdog timer will ensure smoother operation. Other improvements include a debounce to prevent double-click issues, a consistent, graceful stop routine, and connect-time self-tests to verify device communication. Additionally, a persistent safety log will help quickly identify and resolve issues. These updates aim to improve the overall user experience, throughput, and safety of the system.

6. CONCLUSION

We have provided an example of using Python for instrument control in a physics lab, including a graphical user interface for convenient operation by the user.

The programming tasks presented here, such as programming a monochromator, have often been solved in physics labs using LabVIEW. The LabVIEW programming language is excellently suited for hardware integration and real-time control tasks and has established itself as the market leader in this area over the past few decades. A major disadvantage of LabVIEW, however, is the high cost of licensing, especially for smaller projects.

Python has existed since the early 1990s, but its popularity has increased massively only in recent years, and it has become more and more established. Python is a versatile, free software option that is now suitable for a wide range of test automation scenarios.

Although Python has been around since the early 1990s, its popularity has grown significantly in recent years, leading to widespread acceptance. As versatile and free software, it is now well-suited for a wide range of test automation scenarios.

For applications that require precise timing and hardware acceleration, LabVIEW is currently still the superior programming language, but many tasks in hardware-level programming can also be accomplished with Python. Where applicable, this also makes it possible to integrate part of the analysis of the measured data directly into the same program.

With our application for controlling a monochromator, we demonstrated how Python can be used for hardware integration to automate the measurement of spectral data. In combination with computer-controlled tunable lasers, the setup presented here can now be used, for example, for automated measurements with laser-induced fluorescence spectroscopy.

Peer Review: Externally peer-reviewed.

Author Contribution: Conception/Design of study - Ş.Ş, S.K.; Data Acquisition - P.S., C.W.; Data Analysis/Interpretation - Ş.Ş, S.K., C.W.; Drafting Manuscript - S.K., Ş.Ş; Critical Revision of Manuscript - P.S., C.W.; Final Approval and Accountability - S.K., Ş.Ş, P.S., C.W.; Technical or Material Support - C.W., P.S.; Supervision - S.K.

Conflict of Interest: Authors declared no conflict of interest.

Financial Disclosure: Authors declared no financial support.

LIST OF AUTHOR ORCIDS

Ş. Şeker https://orcid.org/0009-0008-5981-4419
P. Schwarz https://orcid.org/0009-0007-6652-2302
C. Weidner https://orcid.org/0009-0008-1274-2382
S. Kröger https://orcid.org/0000-0003-4991-9176

REFERENCES

FAULHABER Drive Systems, Communication / Function Manual, 4th edition, 2014, retrieved from https://www.faulhaber.com/fileadmin/Import/Media/EN_7000_05030.pdf on 07.07.2025

Kramida, A., Ralchenko, Yu., Reader, J. and NIST ASD Team, 2024, NIST Atomic Spectra Database (version 5.12), [Online]. Available: https://physics.nist.gov/asd [on 03.09.2025]. National Institute of Standards and Technology, Gaithersburg, MD.

Kronberger H., 1971, Physics Today, 24, 89, https://doi.org/10.1063/1.3022542

Matplotlib Development Team, Matplotlib – Python plotting, 2025, retrieved from https://matplotlib.org/ on 07.07.2025

National Instruments, 2025a, Spezifikationen des USB-6009. retrieved from https://www.ni.com/docs/en-US/bundle/usb-6009-specs/page/specs.html?srsltid=AfmBOoq113GURNgO0I_6wDwuixtaIH2A5un0Yg-Oh1rmqjSa73BiztwD on 07.07.2025

National Instruments, 2025b, NI-DAQ™mx Python Documentation. retrieved from https://nidaqmx-python.readthedocs.io/ on 07.07.2025

National Instruments, 2025c, NI-DAQ™mx User Manual. retrieved from https://www.ni.com/docs/en-US/bundle/ni-daqmx/page/user-manual-welcome.html on 07.07.2025

National Instruments, 2025d, Queued State Machine (Guide). retrieved from https://learn-cf.ni.com/teach/riodevguide/code/rt_queued-state-machine.html on 07.07.2025

Python Software Foundation, 2025e, `pyserial` — Python Serial Port Extension retrieved from https://pypi.org/project/pyserial/ on 07.07.2025

Python Software Foundation., 2025a, The Python Language Reference. retrieved from https://docs.python.org/3/reference/ on 07.07.2025

Python Software Foundation, 2025b, `concurrent.futures` — Launching parallel tasks. retrieved from https://docs.python.org/3/library/concurrent.futures.html on 07.07.2025

Python Software Foundation, 2025c, `pathlib` — Object-oriented filesystem paths. retrieved from https://docs.python.org/3/library/pathlib.html on 07.07.2025

Python Software Foundation, 2025d, `tkinter` — Python interface to Tcl/Tk. retrieved from https://docs.python.org/3/library/tkinter.html on 07.07.2025

Python Software Foundation, 2025e, `pyserial` — Python Serial Port Extension retrieved from https://pypi.org/project/pyserial/ on 07.07.2025

Şeker, Ş., Kröger, S., Schwarz, P., Weidner, C., 2025, GitHub-Repository-Name [Software] https://github.com/sk-HTW/monochromator_python_code