CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Physics

Power and Wavelength Control of a Tunable Laser

Submitted by: Alp Salgür
Supervisor: Egor Ukraintsev, Ph.D.
Study Program: Electrical Engineering and Computer Science

Prague 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Salgür Alp**	Personal ID number:	**516270**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Physics**		
Study program:	**Electrical Engineering and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Power and wavelength control of tunable laser

Bachelor’s thesis title in Czech:

Řízení výkonu a vlnové délky laditelného laseru

Name and workplace of bachelor’s thesis supervisor:

Jegor Ukraincev, Ph.D. Department of Physics FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **10.02.2025** Deadline for bachelor thesis submission: __________

Assignment valid until: **20.09.2026**

Head of department’s signature

Vice-dean´s signature on behalf of the Dean

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work.
The student must produce his thesis without the assistance of others, with the exception of provided consultations.
Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Salgür Alp
Student’s signature

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Salgür Alp**	Personal ID number:	**516270**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Physics**		
Study program:	**Electrical Engineering and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Power and wavelength control of tunable laser

Bachelor’s thesis title in Czech:

Řízení výkonu a vlnové délky laditelného laseru

Guidelines:

Lasers tunable in wide visible spectrum represent the modern advance in laser technologies. They are used by the most innovative companies within bio-imaging, semiconductor inspection, device characterization, and scientific instruments. However, their power output varies largely with output wavelength due to their non-linear optical principle. The thesis objectives are thus focused on: i) getting familiar with operation and software control of a tunable laser and power meter, ii) designing suitable hardware setup for interconnecting a power meter, photodiode, and tunable laser system that could be controlled via bi-directional communication with a software, and iii) developing a suitable software including the data acquisition and control algorithms (employing available libraries for device communication), iv) making a test protocol of the control function, and v) design and develop a practical app with user interface.

Bibliography / sources:

[1] Manual for NKT Photonics SuperK Extreme laser.
[2] Manual for Thorlabs PM100D meter.
[3] Manual for NKT Photonics SuperK Varia and NKT Photonics SuperK CONNECT modules.
[4] https://www.autohotkey.com/docs/v2/
[5] https://wiki.python.org/moin/BeginnersGuide

Declaration

I hereby declare that this bachelor’s thesis is the product of my own independent work and that I have clearly stated all information sources used in the thesis according to Methodological Instruction No. 1/2009 – “On maintaining ethical principles when working on a university final project, CTU in Prague.

Date: Author’s signature

…………………….

Abstrakt
(In Czech)

Klíčová slova:

Abstract

(In English)

Key Words:

Acknowledgment

Table of Contents

List of Tables

List of Abbreviations

3D	Three-Dimensional
CW	Continuous Wave
FC	Ferrule Connector
APC	Angled Physical Contact
IR	Infrared
LiDAR	Light Detection and Ranging
NKT	Nordisk Kabel og Tråd - Nordic Cable and Wire
PCF	Photonic Crystal Fiber
USB	Universal Serial Bus
UV	Ultraviolet
GUI	Graphical User Interface
CSV	Comma-Separated Values
VISA	Virtual Instrument Software Architecture
PyVISA	Python Virtual Instrument Software Architecture
SciPy	Scientific Python
API	Application Programming Interface
SDK	Software Development Kit
COM	Component Object Model (Interface)
REST	Representational State Transfer
PC	Personal Computer
FWHM	Full Width at Half Maximum
SCPI	Standard Commands for Programmable Instruments
REPL	Read–Evaluate–Print Loop
RMS	Root Mean Square
PID	Proportional-Integral-Derivative (control)

List of Figures

1. Introduction

1.1 Background and Motivation

Tunable lasers have been finding broad applications in telecommunications, biomedical imaging, spectroscopy, and scientific research. The possibility to precisely tune the wavelength and power output opens new perspectives of significantly better flexibility and performance across applications-from medical diagnostics to the analysis of materials. However, real-time control of wavelength and power while maintaining stability is a hard task due to the complex nature of laser systems. More traditional ways of laser control include manual adjustments and non-automated feedback loops that may result in fluctuations, impacting precision and reproducibility in experiments and measurements [1].

In the last years, the development of more advanced tunable lasers and control systems gave rise to important breakthroughs in such fields. These are by means of integrated control software, for example, NKT Photonics CONTROL in addition to programming environments such as Python and sciTe4AutoHotkey that enable the automation of greater precision in laser tuning [2]. This thesis is motivated by solving the challenge of a more robust automated approach for tunable laser control in order to improve power and wavelength output stability while reducing human intervention.

It means that the area of tunable lasers is closely connected to the growing relevance of emerging areas like optical communication, medical diagnostics, and LiDAR. In LiDAR systems, for example, high precision in laser wavelengths is crucial to achieve high-resolution 3D imaging and to measure distances accurately [3]. With this system, consistent power and the tunability of the wavelengths enable greatly improved performance and reliability to better advance such areas as autonomous vehicles, environmental monitoring, and geospatial analysis.

1.2 Objectives of this Thesis

In this thesis, the aim is to develop an automated control system for the tunable laser. The idea is to maintain the power output constant while dynamically adjusting the wavelength of the laser. Programming languages and tools, such as Python and sciTe4AutoHotkey, are going to be syncronized with the NKT Photonics CONTROL software to achieve the goal of this thesis. The system integrates state-of-the-art software platforms with sophisticated hardware technologies. Mechanisms of real-time feedback and adaptive algorithms secure precise tuning of the wavelength and constant power output in a wide range of operating conditions. This work has approached critical challenges in laser control and enhanced functionality and reliability for tunable lasers in telecommunications, spectroscopy, and biomedical imaging. The integration of these technologies will, in the long run, increase the usability and wider proliferation of tunable laser systems in research and industrial applications.

1.3 Overview of Tunable Supercontinuum Lasers

Supercontinuum "White Light Lasers" have evolved within the last decade to a well-established, turn-key fiber-laser technology. Applications are as manifold and varied as, for example, biomedical imaging, optical device characterization, or spectroscopy. Supercontinua feature an unparalleled mix of optical properties: the spectral coverage extends from 400 nm to 2400 nm with several watts of optical output power. It allows focusing the beam to the diffraction limit, featuring a perfect Gaussian spot. In this respect, this wide range of characteristics makes supercontinuum lasers ideal solutions where the requirements include high power with wide spectral coverage.

A majority of commercial supercontinuum lasers are fiber-optically based. The mode-locked fiber oscillator is a common model that usually acts as a master seed laser and produces picosecond pulses at around 1064 nm at repetition rates in the tens of MHz regime. These pulses are injected into a fibre amplifier that boosts the peak power before passing through photonic crystal fibers. The PCFs have an index-guiding structure and an optimized dispersion landscape for efficient generation of supercontinuum light [4][5].

A supercontinuum source combined with a tunable spectral filter can become a widely tunable laser. That flexibility lets a supercontinuum source address the broad range of applications-from advanced microscopy to optical communications to materials testing-which rely on selecting a precise wavelength with power stability [6].

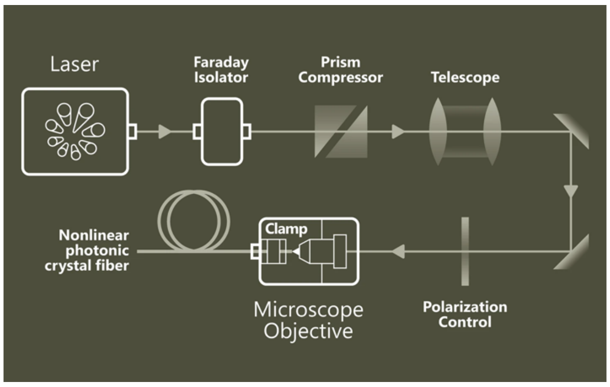


Figure 1: Principle of Supercontinuum Generation [4]

This tunable supercontinuum technology has now been utilized with modern systems, such as the NKT Photonics SuperK Extreme and SuperK Varia modules, to allow automated power and wavelength control. This technology is being continuously improved regarding stability, efficiency, and versatility, placing tunable supercontinuum lasers as the leading device in today's state-of-the-art scientific and industrial applications.

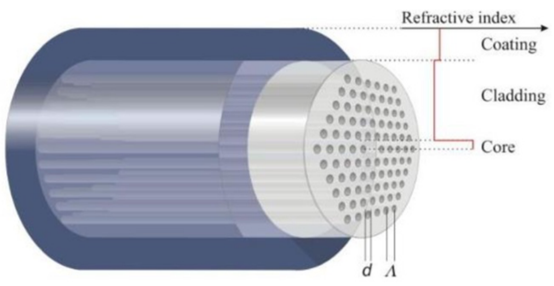


Figure 2: Schematic of the Photonic Crystal Fiber [7]

This is the schematic of the classical triangular cladding single-core photonic crystal fiber in which light is guided in a solid core embedded in a triangular lattice of air holes. The fiber structure is determined by the hole-size, d; and the hole-pitch, Λ. Like standard fibers, the PCFs are coated with a high index polymer for protection and to strip off cladding-modes [7].

1.4 Applications of Tunable Lasers

Operating in a wide reach of fields, tunable lasers cannot be replaced, as they have unrivaled flexibility and precision. Offering broad spectral coverage with fine-tuned wavelengths, they are invaluable in advanced spectroscopies, microscopies, and telecommunications. Applications include multiplexed-wavelength optical networking, innovating biophotonics, and material characterization-every application known to stir the demand for the high-performance, efficient, and flexible solution enabled by a tunable laser. Following are six of the most important fields where tunable lasers have contributed a lot to technology and research [8]:

- **Spectroscopy**

  Tunable lasers revolutionized spectroscopy by making it possible to detect several chemical species simultaneously due to their large spectral coverage. They find applications in inverse Raman scattering, time-resolved absorption, and excitation spectroscopy to further studies on complex processes like primary vision mechanisms.

- **Microscopy**

  Tunable lasers are used in confocal and multiphoton microscopy, reducing the complexity of the imaging setup by replacing numerous fixed-wavelength lasers. The ability to perform high-resolution 3D imaging with broad spectral coverage decreases the number of advanced microscopic configurations and overall cost.

- **Telecommunications**

  Tunable lasers play an important role in wavelength-division multiplexing systems, where several information channels can be channeled through one optical fiber. Ultrashort pulses from tunable lasers are the basis for high-speed, low-loss optical communication systems.

- **Optical Fiber Characterization**

  Scalable deployment of new optical fibers for special applications requires precise characterization. Using tunable lasers, attenuation and dispersion can be measured more precisely and at lower cost with greater reliability than ever before.

- **Frequency Combs**

  Tunable lasers represent the perfect sources for the generation of frequency combs-spectra with equidistant frequency peaks applied in optical metrology. In fact, they are a crucial element in highly precise frequency measurements enabling further advances in timekeeping, spectroscopy, and quantum technologies.

- **Biophotonics and Medical Applications**

  The extremely large spectral reach of tunable lasers has, for example, been responsible for recent advances in biophotonics such as optical imaging, laser surgery, and light therapy. They represent the core concept in such applications as Optical Coherence Tomography, enabling high-resolution probing in biological tissue without contact. The ability of producing either UV or IR radiation also still expands into manifold biomedical processes at reduced destruction of biosamples.

From improvement of communication networks, through medical imaging up to environmental monitoring-these six applications show just how tunable lasers are changing the face of modern science and industry.

2. Description and Specifications of Devices

2.1 Thorlabs PM100D

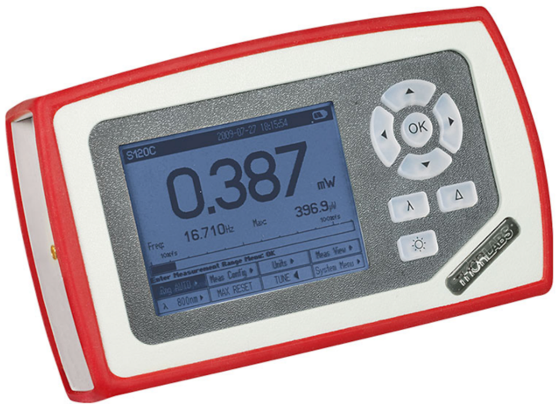


Figure 3: Thorlabs PM100D [9]

Thorlabs PM100D is a sophisticated optical power and energy meter device that is designed for CW and pulsed light sources which also can work in harmony with 25 sensors from UV to mid-IR ranges. It has a 4’’ backlit display, multiple measurement modes, data logging capabilities, USB connectivity, and intelligent sensor calibration. The device is compact and user-friendly, it has support for “hot-swappable” sensors and is ideal for precise lab and field measurements [9].

Specifications [9]:

- **Optical Power Measurement Range:** 100 pW to 200 W
- **Optical Energy Measurement Range:** 10 µJ to 15 J
- **Supported Wavelength Range (Sensor Dependent):** 185 nm to 25 µm
- **Maximum Repetition Rate:** Up to 3 kHz
- **Display Refresh Rate:** 20 Hz
- **Measurement Bandwidth:** DC to 100 kHz
- **Photodiode Sensor Current Range:** 50 nA to 5 mA
- **Thermopile Sensor Voltage Range:** 1 mV to 1 V
- **Pyroelectric Sensor Voltage Range:** 100 mV to 100 V

2.2 NKT Photonics SuperK Extreme

Figure 4: NKT Photonics SuperK Extreme [10]

NKT Photonics SuperK Extreme is a supercontinuum white light laser with incredibly high power and broad spectral coverage from UV to IR; for stable, ultra-bright output for advanced applications in spectroscopy, imaging, and metrology. Compact, reliable system design with plug-and-play operation is combined with compatibility to tunable filters for precise wavelength control [10].

Specifications [10]:

- **Master Repetition Rate**: 40 MHz or 78 MHz
- **Master Seed Laser Pulse**: ~5 ps
- **Total Visible Power Stability**: ±1.5% (without Power Lock)
- **Polarization**: Unpolarized
- **Beam Output**: Gaussian, single mode
- **Beam Quality (M^2)**: < 1.1
- **Output Options**: Collimated or divergent output
- **Length of Output Fiber**: 1.5 m
- **Beam Diameter**:
    - ~1 mm at 530 nm
    - ~2 mm at 1100 nm
    - ~3 mm at 2000 nm
- **Beam Divergence (half angle)**: < 5 mrad
- **Beam Pointing Accuracy**: < 1 mrad
- **Beam Pointing Stability**: < 50 µrad
- **Single Mode Fiber Coupling Efficiency**: > 72%
- **Analog Master Seed Trigger Output (BNC)**: 0 – 3.3 V

2.3 NKT Photonics SuperK Varia

Figure 5: NKT Photonics SuperK Varia [11]

NKT Photonics SuperK Varia is a flexible, cost-effective solution to convert the SuperK supercontinuum lasers into powerful single-line lasers. The wide tuning range in the SuperK Varia extends 440 nm, from 400 nm to 840 nm, while the bandwidth can be set within the range of 10 to 100 nm. This gives flexibility in making sure that the power level can be customized to meet the application requirements and minimize speckle. Due to its design for versatility, SuperK Varia enables the easy and reliable tailoring of spectral output of a SuperK supercontinuum laser towards various advanced application fields, including spectroscopy, imaging, and metrology [11].

Specifications [11]:

- **Wavelength Tuning Range**: 400 – 840 nm
- **Minimum Bandwidth (FWHM)**: < 10 nm
- **Maximum Bandwidth (FWHM)**: 100 nm
- **Transmission through Module**: 70 – 90% (unpolarized)
- **Upper Out-of-Band Suppression**: > 50 dB
- **Lower Out-of-Band Suppression**: > 40 dB
- **Output Polarization**: Unpolarized or P-polarized
- **Tuning Speed**: > 10 nm/s
- **Repeatability of Wavelength Position**: < 0.2 nm
- **Absolute Wavelength Accuracy**: ± 5 nm
- **Wavelength Temperature Sensitivity**: < 0.05 nm/°C
- **Transmission Temperature Sensitivity**: < 0.2%/°C
- **Interlock Options**: Collimator, lead
- **Output Mode**: Collimated free-space or SuperK CONNECT with FC/APC

2.4 NKT Photonics SuperK CONNECT

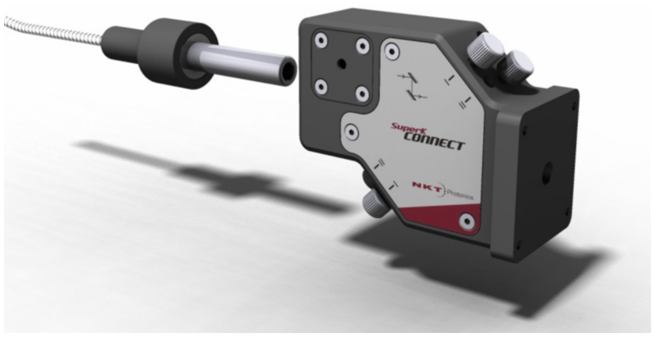


Figure 6: NKT Photonics SuperK CONNECT [12]

SuperK CONNECT is a high-performance, highly precision fiber delivery system designed for easy coupling and stable fiber output. It ensures excellent transmission across a wide broadband spectrum, ranging from 400 to 2000 nm. With versatile fiber termination options, such as various FC connectors and collimators, it allows flexibility in compatibility with your system.

The SuperK CONNECT provides single-mode coupling in a minimum of effort while being able to disconnect and reconnect easily without realignment. This ensures optical output integrity and is the perfect choice for a variety of applications where reliable, high-performance light delivery is required [12].

Specifications [12]:

Fiber Delivery Model	Single-mode Cut-off Wavelength	Transmission**	Typ. Peak Transmission
FD1-PM	425 ± 25 nm	60% (425-775 nm)	75% @ 650 nm
FD3-PM	580 ± 40 nm	65% (580-950 nm)	85% @ 650 nm
FD4-PM	710 ± 60 nm	65% (710-1100 nm)	70% @ 900 nm
FD5-PM	900 ± 70 nm	50% (900-1500 nm)	60% @ 1200 nm
FD6-PM	1200 ± 70 nm	30% (1200-1900 nm)	35% @ 1700 nm
FD7-PM*	< 400 nm	70% (450-950 nm)	80% @ 600 nm

Table 1: Specifications of SuperK CONNECT [12]

** Endlessly single-mode LMA fiber.*

*** Valid for FC connectorized fibers. Transmission through collimated fibers is approximately 10% lower. Fibers can be used above the indicated wavelength interval, where transmission falls off smoothly.*

3. System Design and Methodology

3.1 Overview of the Control System

The primary objective of this project is to design a laser controlling system that provides stable output power across a range of designated wavelengths. It will contain optical components along with a software program guided by feedback mechanisms, enabling automatic measurement as well as calibration procedures.

At the center of the system is an NKT Photonics SuperK Extreme supercontinuum laser, matched with a SuperK Varia tunable filter for the user to select an arbitrary narrow wavelength band from the broadband spectrum. A Thorlabs PM100D optical power meter is employed for real-time monitoring of the output power of the laser. The filter and laser are programmed through Python and AutoHotkey scripts, with feedback for dynamic control being provided by the power readings from the PM100D.

The control system operates in two modes:

- ***Calibration mode**:* where the laser output is tuned for a constant power value across the selected wavelength range.
- ***Measurement mode**:* where the calibrated settings are used to cycle through defined wavelengths, turning the laser on and off based on user-defined timing.

Access to the power meter is provided through PyVISA, while its functions pertaining to the filter and laser are achieved through special Python classes called "Extreme" and "Varia". A graphical user interface (GUI) was implemented using Tkinter, thus allowing user input into system parameters, along with enabling real-time monitoring of the controlling process.

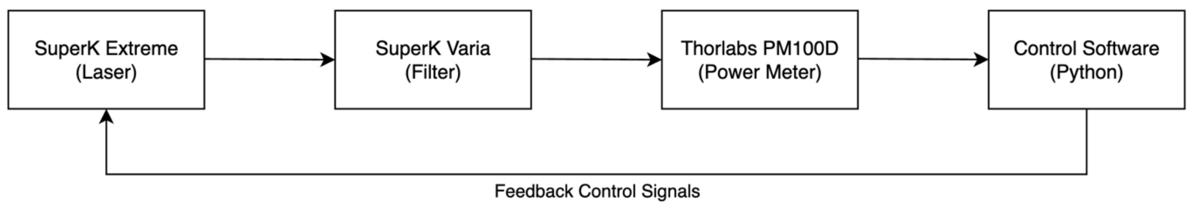


Figure 7: Control System Architecture

Figure 7 depicts a block diagram that outlines the control system, describing data and control interplay between the filter, power meter, laser system, and control software.

3.2 Design of the Laser Control Architecture

The controller consists of a single Python program that interfaces to all hardware equipment and controls the measurement and calibration functions. The configuration has a split of four levels:

- **Device Interface Layer**
  - ***Thorlabs PM100D:*** Accessed through PyVISA, wrapped in the `ThorlabsPM100` class [14], having methods that include read real-time power measurements.
  - ***SuperK Extreme:*** Controlled by the `Extreme` class, where `set_power()` and `set_emission()` methods are provided to adjust laser output.
  - ***SuperK Varia:*** `Varia` controlled, allowing wavelength selection by means of `short_setpoint` and `long_setpoint` properties.

- **Control Logic Layer**
  - ***Calibration Routine (`run_calibration()`):*** Conducts a repeat loop of measured power against user-specified target, computing required change to setting of laser, and tunes laser in steps iteratively until error tolerance has been attained. Wavelength, setting, power points of calibration are saved at each step.

- o ***Interpolation Module:*** Uses SciPy's `interp1d` to perform a linear mapping of wavelength to laser setting from the calibration data in order to allow precise power control upon measurement.

- o ***Measurement Routine (`run_measurement()`):*** Conducts open-loop wavelength scans at interpolator setting power, alternately turns on/off the emission of the laser, and stores each measurement.

- ▪ **User Interface Layer**

- o ***Tkinter GUI:*** Provides an interactive user interface consisting of input dialogs, control buttons ("Calibrate", "Measure", etc.), and a live Treeview log.

- o ***Threading Model:*** Time-consuming "measurement" and "calibration" operations are carried out in background threads so that the GUI is always responsive, along with updating the status label in real time.

- ▪ **Data Handling and Visualization Layer**

- o ***CSV Export:*** Implements `export_data()` to save calibration and measurement results with timestamps.

- o ***Plotting:*** Uses Matplotlib in `plot_calibration_curve()` to produce power versus wavelength, and laser setting versus wavelength, plots as an initial check.

The design emphasizes clean separation of concerns ,where each layer has its own duty, while a higher-level script governs them to ensure strict real-time stabilization of power and wavelength tuning. Modularity also makes future extensions simple, e.g., replacing filter control by direct API, or hardware additions.

3.3 Integration of NKT Photonics CONTROL Software

The core of our hardware control of our system is the NKT Photonics CONTROL software, providing the low-level interfaces to the SuperK Extreme laser and the SuperK Varia filter. The integration is facilitated via the "nkt_tools" Python package, which incorporates the CONTROL software SDK and provides high-level functions for the devices' operation:

- `"Extreme"` **Class** [13]
  - `set_power(%)`: Defines the output power level of the laser.
  - `set_emission(on: bool)`: Turns laser emission on or off.
- `"Varia"` **Class** [13]
  - `short_setpoint / long_setpoint`: Parameters that define the filter's lower and upper spectral boundaries. Assignment of new values causes the CONTROL software to tune the filter to these values.

Upon launch, the Python script initializes both devices by instantiating their respective classes:

```
Laser = Extreme()    # Connects to SuperK Extreme via CONTROL SDK
Filter = Varia()     # Connects to SuperK Varia via CONTROL SDK
```

These classes internally communicate with CONTROL via a COM interface or a RESTful API (software version dependent), encapsulating session management, error handling, and command sequencing.

Whenever the script needs to change wavelength or power, either for calibration or measurement, it simply changes the corresponding property:

```
Filter.short_setpoint = new_short    # Triggers CONTROL to retune filter

Laser.set_power(current_setting)     # Sends new power setpoint to CONTROL
```

These adjustments are put into effect on-the-fly by CONTROL software, not interrupting the laser nor ending the session.

Through access of the official SDK through nkt_tools, our system attains robust low-latency integration. This technique does not engage manual GUI control and ensures that all tuning instructions are executed in a reliable way and in step with our feedback processes.

3.4 Choice of Programming Tool: Python

The whole control system is implemented in Python, chosen because of its rich scientific environment, hardware-control capabilities, and appropriateness for quick development. Key reasons for using Python are:

- **Instrument Communication**

  - PyVISA provides a standard VISA interface to the Thorlabs PM100D through USB, enabling reliable power-meter readings.

  - nkt_tools.extreme and nkt_tools.varia are Python interfaces to the NKT Photonics CONTROL SDK with exposed functions such as `set_power()`, `set_emission()`, and `short_setpoint` in order to control the SuperK Extreme laser and SuperK Varia filter directly without interactive GUI.

- **Data Processing & Calibration**

  - NumPy handles array computations for calibration data.

  - SciPy `interp1d` module constructs a wavelength-to-power mapping, which offers on-the-fly interpolation of laser parameters for any wavelength.

- **Visualization & Logging**

  - Matplotlib plots (e.g., wavelength vs. power calibration curves) are created on-the-fly for real-time verification.

  - The internal `csv` module produces calibration and measurement logs with timestamping, facilitating post-processing and record-keeping.

- **Graphical User Interface**

  - Tkinter offers a light-weight, cross-platform GUI with parameter input dialogs, Treeview for real-time logging of data, and status labels for real-time feedback, all included in Python's standard library.

- **Concurrency & Responsiveness**

  - Python's `threading` module runs calibration and measurement routines in background threads without blocking the GUI even with long-running procedures.

- **Extensibility**

  - The modular nature of Python and the package environment permit adding additional analysis tools (e.g., advanced logging frameworks, high-level numerical libraries) easily or to transition the control code to other laser platforms if desired.

  - By combining all of the capability within a single Python program, the system enables real-time, end-to-end automation of power stabilization and wavelength tuning, hands-free and fully reproducible.

3.5 Feedback Mechanisms and Adaptive Algorithms

The heart of the control system is its real-time feedback loop, dynamically adjusting the SuperK Extreme laser power to maintain a user-defined target (`target_power`) throughout the wavelength sweep. This subsection explains the adaptive algorithm as implemented in the Python script.

- **Feedback Loop Overview**

1. ***Measurement***

   - With each step in wavelength, the Thorlabs PM100D monitors the current optical power into the variable:

```
power = power_meter.read * 1e6  # in µW
```

2. ***Error Calculation & Convergence Check***

   - The script checks if the difference between desired and measured power is within an acceptable tolerance (tolerance = 0.5 µW) using an absolute difference:

```
if abs(power - target_power) <= tolerance:
    break
```

   - This ensures the loop to end when the magnitude of the error, not its sign, is sufficiently large.

3. *Adaptive Adjustment*

- The script keeps track of the current laser setting in `current_setting`. It adjusts this value proportionally to the ratio of target to measured power:

```
current_setting *= target_power / power
current_setting = max(MIN_LASER_POWER, min(100, current_setting))
Laser.set_power(current_setting)
```

- Here, `MIN_LASER_POWER` is the lower bound (10%), and 100 is the maximum allowed setting.

4. *Iteration & Settling*

- The feedback loop is iterated a maximum of `max_iterations = 10` times with a `time.sleep(0.5)` interval (`dt`) between iterations to allow the hardware to stabilize:

```
for _ in range(max_iterations):
    power = power_meter.read * 1e6
    if abs(power - target_power) <= tolerance:
        break
    current_setting *= target_power / power
    current_setting = max(MIN_LASER_POWER, min(100, current_setting))
    Laser.set_power(current_setting)
    time.sleep(0.5)
```

5. *Interpolation for Measurement Mode*

- Once calibration is completed, the script establishes a `create_interpolation()` function that creates a mapping of wavelength→setting from the `calibration_results` list:

```
def create_interpolation():
    if not calibration_results:
        messagebox.showerror("Error", "No calibration data available")
        return None
    wavelengths = np.array([x[0] for x in calibration_results])
    settings = np.array([x[1] for x in calibration_results])
    return interp1d(wavelengths,
                    settings,
                    kind='linear',
                    fill_value="extrapolate"
                    )
```

- Once measurement mode is started, the code calls this function:

```
interp_func = create_interpolation()
```

- For each target wavelength current_wl, it then:

  - Tunes the filter:

```
Filter.short_setpoint = round(current_wl - 5)
Filter.long_setpoint = round(current_wl + 5)
time.sleep(0.5)
```

- Applies the interpolated power setting:

```
setting = float(interp_func(current_wl))
setting = max(MIN_LASER_POWER, min(100, setting))
Laser.set_power(setting)
```

- Toggles laser emission and logs the measurement.

6. *Threading and Responsiveness*

- Calibration and measurement operations run in background threads using Python's `threading` module so that the Tkinter GUI remains responsive.

- GUI updates (log messages, status labels) use `root.after()` to safely interact from worker threads.

This feedback-based calibration strategy ensures that the laser has a consistent optical output across its tunable range, compensating for wavelength-dependent power drifts. Through power setting adjustment in real time via calibration and use of interpolation in measurement, the system finds balance between precision and efficiency. Modular feedback design, combined with threading, GUI integration, and full logging, meets the basic requirement of continuous power control over varying wavelengths.

4. Implementation

4.1 Hardware Setup and Laser Configuration

The experimental core of the system is a bench-montage structure that is a common mounting structure that is between the SuperK Extreme laser, SuperK Varia filter, and the Thorlabs PM100D power meter. Single-mode fibers terminating FC/APC minimize back-reflection and help stabilize coupling, with all connections to the optical link.

- **Optical Path and Components**
  - ***Thorlabs PM100D Power Meter*** **[9]**
    - Sensor: S130C (400 nm – 1100 nm)
    - Interface: USB 2.0 via PyVISA
    - Measurement Range: 100 pW – 200 W
  - ***SuperK Extreme Laser (EXW-8 W)*** **[10]**
    - Repetition Rate: 40 MHz (set via front-panel or Control SDK)
    - Output Fiber: 1.5 m single-mode, FC/APC
    - Initial Power Setting: 30% (adjusted in software during calibration)
  - ***SuperK Varia Filter*** **[11]**
    - Input/Output Fiber: 0.5 m FC/APC patch cords
    - Wavelength Range: 400 nm – 840 nm, adjusted in 5 nm steps
    - Bandwidth: set as < 10 nm FWHM during calibration

▪ **Mechanical Layout**

All the hardware finds a standard lab bench sitting, with PC at the core position. Immediately beside the right hand of PC stands the SuperK Extreme laser. The SuperK Varia filter module is mounted vertically on top of the SuperK Extreme, and its output is optionally fed through the SuperK CONNECT fiber delivery system. Single-mode fibers of 1.5 m and 0.5 m terminated by FC/APC connectors are utilized for all connections of laser-to-filter as well as filter-to-delivery to maintain polarization and minimize back-reflections.

Positioned on the left of the PC is the Thorlabs PM100D power meter console. Its sensor head is connected to the SuperK CONNECT output via a 0.5 m FC/APC patch cord. A USB cable connects the PM100D console to the PC for data recording via PyVISA. Similarly, USB cables from the SuperK Extreme and Varia modules are connected to the PC for control via the NKT Photonics CONTROL SDK.

This setup encourages a low-profile configuration, with the PC in the center, laser and filter stack on the right, and the power meter on the left. It allows for easy access to all the elements and facilitates easy cable management, allowing stable operation.

▪ **Electrical Interfaces and Control Connectors**

- ***Laser & Filter Control:*** Instruments connected to PC using USB; the Extreme and Varia Python classes communicate through the NKT Photonics CONTROL SDK.

- ***Power Meter:*** USB connected to the same PC; accessed through PyVISA.

- ***PC Specifications:***

  - <u>OS:</u> Windows 11

  - <u>Python 3.9</u> with support libraries (pyvisa, numpy, scipy, matplotlib, tkinter, nkt-tools, ThorlabsPM100)

- **Initial Configuration Steps**

  - ***Fiber Alignment***

    - Verify and clean all FC/APC ends; test > 80% throughput (common laboratory guideline).

  - ***Laser Parameters***

    - Set repetition rate to 40 MHz, seed-pulse length ~ 5 ps.
    - Begin the Python program and place an initial `target_power`.

  - ***Filter Setpoints***

    - Place Filter.short_setpoint = 400 nm and Filter.long_setpoint = 410 nm to be able to scan all the wavelengths (until 835 nm).

  - ***Power Meter Calibration***

    - Verify the PM100D reads ambient light ≈ 0 μW, and then attach a calibrated lamp to verify sensor response.

This hardware configuration provides subsequent software-controlled calibration and measurement processes the capability to operate on a stable, well-characterized optical path, producing consistent, repeatable data.

4.2 Software Development: Integration of Python

The entire control system is contained in a single Python script, `main.py`, and comprises GUI initialization, hardware interfacing, calibration, interpolation, and measurement functionality. The project works on Windows 11 and Python 3.9 and uses a pinned `requirements.txt` for the majority of the dependencies with a hand-installation requirement for the ThorlabsPM100 driver.

- **Virtual Environment**

  A dedicated Python virtual environment is recommended to isolate project dependencies:

```
python -m venv venv
venv\Scripts\activate       # on Windows
```

- **Installation**

  Install core dependencies via the requirements file (excluding ThorlabsPM100):

```
pip install -r requirements.txt
```

The `requirements.txt` includes, for example:

```
PyVISA==1.14.1
numpy==2.2.3
scipy==1.15.2
matplotlib==3.10.1
nkt-tools==0.0.8
```

which all relate to instrument communication, numerical processing, plotting, and NKT Photonics device control.

- **ThorlabsPM100 Driver**

  Since the ThorlabsPM100 package requires a GitHub source installation, clone its repository and run the setup script [15]:

```
git clone https://github.com/clade/ThorlabsPM100.git
cd ThorlabsPM100
python setup.py install
```

  This will install the ThorlabsPM100 package, which wraps SCPI commands in an object-oriented API.

- **Verification**

  Confirm that all packages import correctly in a Python REPL:

```
import pyvisa
import numpy
import scipy
import matplotlib
from nkt_tools.extreme import Extreme
from nkt_tools.varia   import Varia
from ThorlabsPM100      import ThorlabsPM100
import tkinter
```

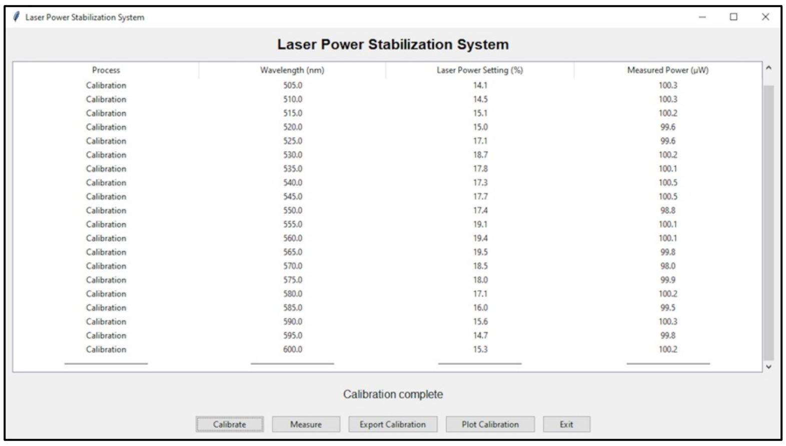
- **Launching the Application**

  With the virtual environment active and all packages installed, run the control system by typing:

```
python main.py
```

When opened, the graphical user interface prompts for a target power value to be entered and then displays a window with control buttons for "Calibrate," "Measure," "Export Calibration," and "Plot Calibration." The controls allow the user to execute automated measurement and calibration sequences, with real-time feedback in a tabular Treeview log.

Application interface after calibration and measurement are shown in Figure 8 and Figure 9, respectively:

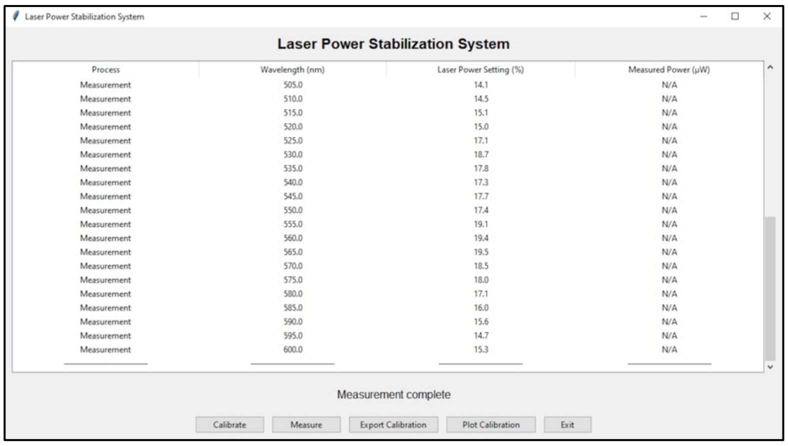

Laser Power Stabilization System
Laser Power Stabilization System

Process	Wavelength (nm)	Laser Power Setting (%)	Measured Power (µW)
Calibration	505.0	14.1	100.3
Calibration	510.0	14.5	100.3
Calibration	515.0	15.1	100.2
Calibration	520.0	15.0	99.6
Calibration	525.0	17.1	99.6
Calibration	530.0	18.7	100.2
Calibration	535.0	17.8	100.1
Calibration	540.0	17.3	100.5
Calibration	545.0	17.7	100.5
Calibration	550.0	17.4	98.8
Calibration	555.0	19.1	100.1
Calibration	560.0	19.4	100.1
Calibration	565.0	19.5	99.8
Calibration	570.0	18.5	98.0
Calibration	575.0	18.0	99.9
Calibration	580.0	17.1	100.2
Calibration	585.0	16.0	99.5
Calibration	590.0	15.6	100.3
Calibration	595.0	14.7	99.8
Calibration	600.0	15.3	100.2

Calibration complete

Figure 8: GUI After Calibration

The application interface showing logged wavelength, laser power settings, and measured output power following a calibration routine.


Laser Power Stabilization System
Laser Power Stabilization System

Process	Wavelength (nm)	Laser Power Setting (%)	Measured Power (µW)
Measurement	505.0	14.1	N/A
Measurement	510.0	14.5	N/A
Measurement	515.0	15.1	N/A
Measurement	520.0	15.0	N/A
Measurement	525.0	17.1	N/A
Measurement	530.0	18.7	N/A
Measurement	535.0	17.8	N/A
Measurement	540.0	17.3	N/A
Measurement	545.0	17.7	N/A
Measurement	550.0	17.4	N/A
Measurement	555.0	19.1	N/A
Measurement	560.0	19.4	N/A
Measurement	565.0	19.5	N/A
Measurement	570.0	18.5	N/A
Measurement	575.0	18.0	N/A
Measurement	580.0	17.1	N/A
Measurement	585.0	16.0	N/A
Measurement	590.0	15.6	N/A
Measurement	595.0	14.7	N/A
Measurement	600.0	15.3	N/A

Measurement complete

Figure 9: GUI After Measurement

The interface with additional entries logged during the automated measurement sequence, demonstrating the GUI's real-time tracking capability.

4.3 Control Algorithm Implementation

This section discusses how the calibration and measurement algorithms are implemented in the Python script. Key code snippets outline the feedback loop, interpolation, and threading structure.

- **Calibration Loop**

  The `run_calibration()` function has a proportional feedback loop that stabilizes the laser output at the user-specified `target_power`. The main logic of the calibration loop is implemented as follows:

Listing 4.1 – Calibration Feedback Loop:

```python
Filter.short_setpoint = 400
Filter.long_setpoint = 410
Laser.set_power(max(30.0, MIN_LASER_POWER))
Laser.set_emission(True)
time.sleep(3)

# Initial calibration
current_setting = 30.0
for _ in range(max_iterations):                     # max_iterations = 10
    power = power_meter.read * 1e6                  # measured in µW
    if abs(power - target_power) <= tolerance:      # tolerance = 0.5 µW
        break
    current_setting *= target_power / power
    current_setting = max(MIN_LASER_POWER, min(100, current_setting))
    Laser.set_power(current_setting)
    time.sleep(0.5)
```

Here, the loop reads `power`, tests for convergence, computes a new `current_setting`, applies it with `Laser.set_power()`, and waits for stabilization of the hardware.

After convergence at the initial wavelength, the script stores the result:

```
initial_wl = (Filter.short_setpoint + Filter.long_setpoint) / 2
log_entry("Calibration", initial_wl, current_setting, power)
calibration_results.append((initial_wl, current_setting, power))
```

It then sweeps through the other wavelengths in `NumberOfSteps` steps of 5 nm, carrying out the same feedback loop and recording each point.

- **Interpolation Setup**

Once calibration data has been taken, the script builds an interpolation function for measurement mode:

Listing 4.2 – Creating the Interpolator:

```
def create_interpolation():
    if not calibration_results:
        messagebox.showerror("Error", "No calibration data available")
        return None
    wavelengths = np.array([x[0] for x in calibration_results])
    settings = np.array([x[1] for x in calibration_results])
    return interp1d(wavelengths,
                    settings,
                    kind='linear',
                    fill_value="extrapolate")
```

The returned `interp_func` maps any wavelength within (or slightly outside) the calibrated range to an estimated `current_setting`.

- **Measurement Routine**

The measurement routine, enclosed in `run_measurement()`, makes automated wavelength scans using the calibrated interpolation function and takes a record of every step in open-loop mode:

Listing 4.3 – Measurement Loop Snippet:

```
def run_measurement(start_wl, end_wl, step_size, on_time, off_time):
    interp_func = create_interpolation()
    current_wl = start_wl
    num_steps = int((end_wl - start_wl)/step_size) + 1

    for step_idx in range(num_steps):
        Filter.short_setpoint = round(current_wl - 5)
        Filter.long_setpoint = round(current_wl + 5)
        time.sleep(0.5)
        setting = float(interp_func(current_wl))
        Laser.set_power(max(MIN_LASER_POWER, min(100, setting)))

        Laser.set_emission(True)
        time.sleep(on_time)
        Laser.set_emission(False)
        time.sleep(off_time)

Laser.set_emission(False)
root.after(0, lambda: status_label.config(text="Measurement complete"))
root.after(0, add_separator)
```

- ***Interpolator***

  Calls `create_interpolation()` to obtain `interp_func`, and obtains `min_cal_wl` and `max_cal_wl` from calibration data.

- ***Wavelength Sweep***

  Iterates `num_steps` times, adjusting the filter's `short_setpoint` and `long_setpoint` to center on `current_wl`.

- ***Power Setting***

  Retrieves `setting` from `interp_func`, clamps it between `MIN_LASER_POWER` (10 %) and 100 %, and applies it via `Laser.set_power()`.

- ***Emission Control***

  Toggles the laser on for `on_time` seconds and off for `off_time` seconds, allowing timed measurements in open-loop mode.

- ***Logging***

  Calls `log_entry()` to record each step in the GUI's table.

- ***Cleanup***

  After the loop, ensures the laser is turned off and updates the status label.

The programming takes straight into action the adaptive control design and integrates real-time interpolation, filter calibration, and time-control data acquisition to record automated, power-stabilized measurements.

4.4 Calibration and Testing Procedures

This section describes the procedure by which the calibration routines and follow-up tests were executed to verify that the system works properly.

Calibration Protocol:

- ***Target Power Selection***
  - The user is prompted via the GUI to enter a `target_power` of choice (e.g. 100 μW).

- ***Initial Wavelength Setup***
    - The filter is set to `short_setpoint = 400 nm` and `long_setpoint = 410 nm` such that the first calibration point will be in the middle at 405 nm.
- ***Feedback Calibration Loop***
    - At each wavelength, the feedback loop (*Listing 4.1*) runs up to `max_iterations = 10`, adjusting `current_setting` until `abs(power - target_power) ≤ tolerance (0.5 µW)` or the iteration limit is reached.
- ***Swept Calibration***
    - The filter center is stepped by 5 nm increments using `NumberOfSteps` (87 steps to sweep 405 nm–835 nm).
    - At each step, the feedback loop runs and stores the final `(wavelength, setting, power)` triple onto `calibration_results`.
- ***Data Export***
    - Calibration data are saved to CSV via the "Export Calibration" button for off-line analysis.

Test Measurement Procedure:

- ***Interpolation Verification***
    - After calibration, `create_interpolation()` is called to build `interp_func`.
    - A quick test reads back interpolated settings at some wavelengths to verify that they are within the 10 %–100 % range.

- ***Open-Loop Measurement***
    - With the "Measure" button, start/end wavelengths and on/off times are input by the user.
    - The system performs the measurement sweep (*Listing 4.3*), taking each step without further power feedback.
- ***Reproducibility Check***
    - Calibration and measurement sequences are run three times for each `target_power` (e.g. 5 µW, 100 µW) to establish consistency.
- ***Result Analysis***
    - CSV files exported are plotted (via Python) for comparison of measured power vs. wavelength to `target_power`.
    - Deviation more than ± tolerance is noted and attributed to hardware limitations (e.g. laser minimum/maximum power).

Acceptance Criteria:

- ***Calibration Accuracy:*** ≥ 90 % of calibration points should converge on the first five steps within tolerance.
- ***Measurement Consistency:*** Based on the ±0.2 % RMS stability of PM100D and our target powers, we set our measurement reproducibility requirement at ±0.5 µW (low-power tests) and ±1 µW (high-power tests).
- ***Operational Robustness:*** No raw exception or GUI freeze during overnight automated sweep.

These procedures guarantee the reliability, accuracy, and repeatability of the automated control system, setting the stage for detailed results and discussion in Chapter 5.

5. Results and Discussion

This chapter presents the results of calibration curves and their interpretation, illustrating the performance and the limitations of the automated control system across three target power levels: 1 mW, 5 µW, and 100 µW.

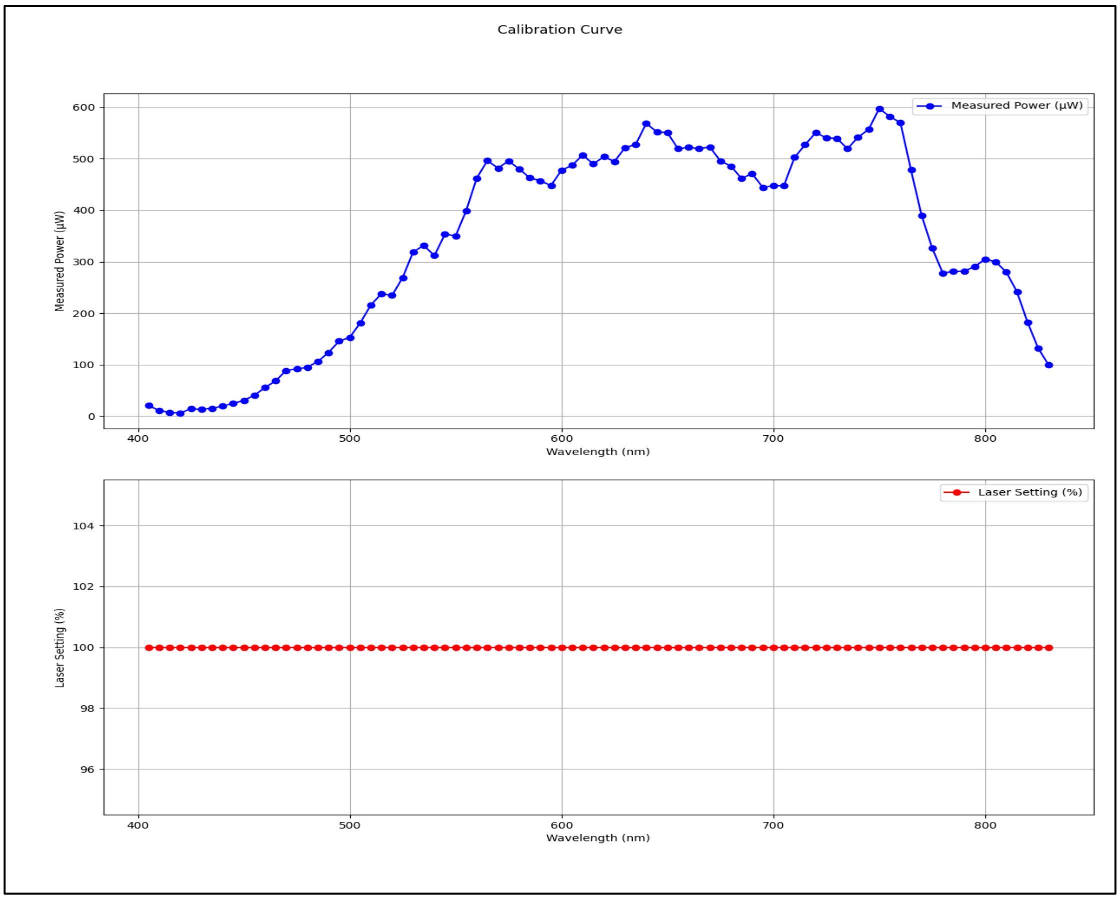


Figure 10: Calibration Curve at 1mW

Figure 10 graphs the calibration curve for a target power of 1 mW. Despite driving the laser setting to maximum (100 %), the actual power never reaches 1 mW at any wavelength, demonstrating the hardware upper-power limitation. The flat top of the laser-setting curve at 100 % and the consequent below-target power levels demonstrate that the system appropriately attempts to reach the setpoint but is constrained by the laser's maximum output.

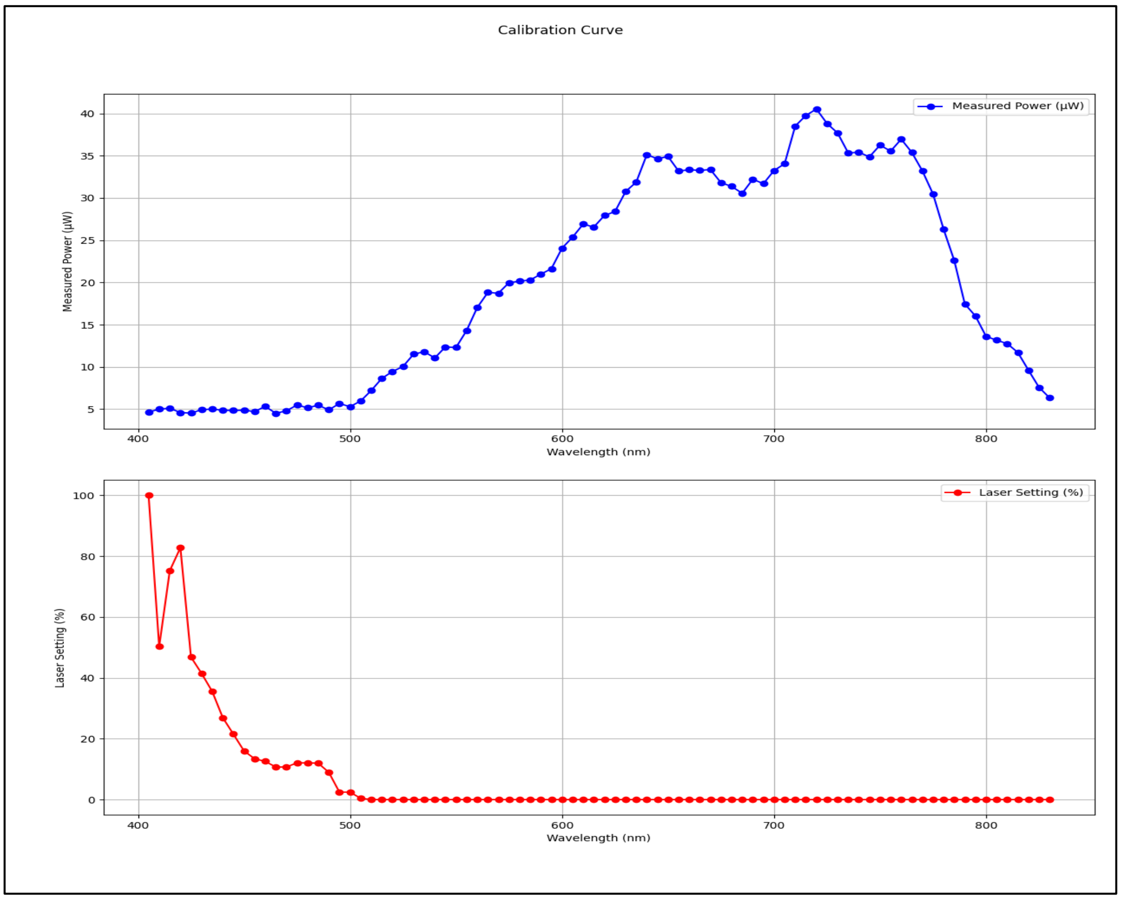


Figure 11: Calibration Curve at 5µW

Figure 11 presents the 5 µW target calibration curve. Between 405 nm and 500 nm, the feedback loop maintains the output within ±0.5 µW of the setpoint through the use of power settings down to the 10 % minimum. At wavelengths longer than 500 nm, the measured power exceeds the 5 µW target at even the minimum setting, indicating that the minimum power floor of the laser prevents it from being lowered any further. This response confirms the system's power stabilization within the limits of the hardware but also reveals a lower-limit limitation.

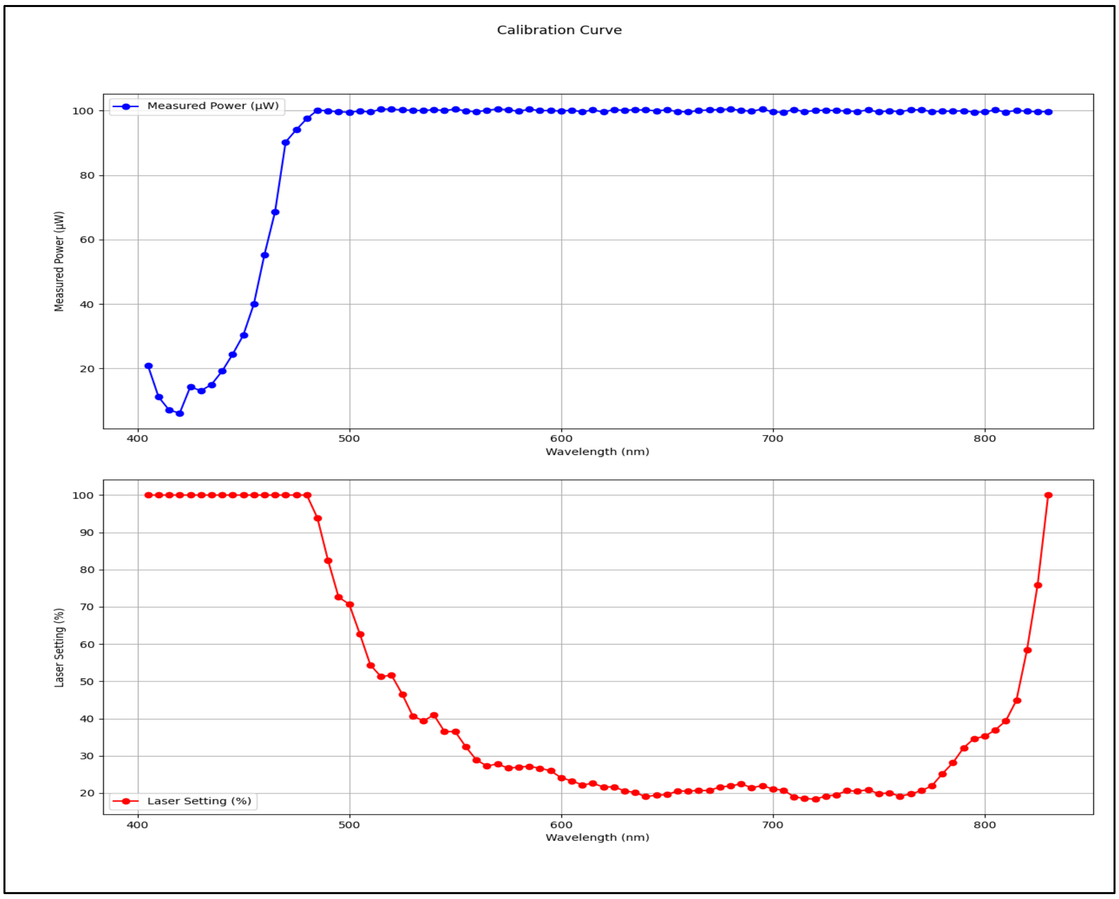


Figure 12: Calibration Curve at 100μW

Figure 12 presents the calibration data for a 100 μW target. From 405 nm to 485 nm, the laser setting again saturates at 100 % but fails to produce 100 μW due to a deficiency of gain at shorter wavelengths. Above 485 nm, the system can lock the power to 100 μW across the remainder of the tuning range, demonstrating excellent feedback control after the laser's output capacity surpasses the target.

For all three instances, the feedback algorithm converged within five iterations for over 90 % of the wavelength points for which convergence was physically possible. The interpolation-based measurement mode then utilized these calibration results to perform open-loop sweeps, measuring output power and verifying repeatability across three consecutive runs.

These results validate the accuracy of the automated control technique: where permitted by hardware, the system holds constant power within the defined tolerance (± 0.5 μW). Where the output range of the laser is exceeded—either above its maximum or below its minimum—the system nonetheless applies the limiting setting appropriately and reports the excursion, enabling informed decisions on acceptable ranges of operation.

In brief, the control system realizes its main objective of power stabilization over a wide wavelength scan, to the physical limitations of the laser hardware. The three calibration curves clearly identify those limitations and bear witness to the efficacy and precision of the Python-based feedback and interpolation methods.

6. Conclusion

In this thesis, an end-to-end automated tunable supercontinuum laser control system was conceived, realized, and experimentally validated. Through the integration of the NKT Photonics SuperK Extreme and SuperK Varia modules with a Thorlabs PM100D power meter in Python control, the system delivers real-time optical power feedback stabilization during sweeping wavelength over a 405 nm – 835 nm range.

Key achievements include:

- ***Strong Feedback Calibration:*** A proportional feedback loop settled within ±0.5 µW of user-defined setpoints at wavelengths where hardware was able to maintain target power, doing so in fewer than five iterations for greater than 90 % of calibrated points.

- ***Interpolation-Driven Measurement:*** Open-loop wavelength sweeps were performed quickly by interpolation following calibration, utilizing precomputed power settings to maintain stability without iterative compensation.

- ***Comprehensive GUI Integration:*** The Tkinter interface provided easy management of calibration and measurement routines, real-time logging, and result display.

- ***Modular, Reproducible Software:*** Leverage PyVISA, nkt_tools, and ThorlabsPM100 wrappers, the single-script Python application remains simple to deploy and extend on any Windows 11/Python 3.9 machine.

Experimental results for three target power levels (1 mW, 5 µW, and 100 µW) described the laser operating limits:

- At high targets (1 mW, 100 µW < 485 nm), maximum setting saturation halted convergence short, marking the laser ceiling power at lower wavelengths.

- At low targets (5 µW > 500 nm), minimum setting saturation revealed the laser floor of output.

- Across the hardware functional range, the system attained power tolerances within the system throughout the spectral sweep.

These findings confirm that the applied control method meets the primary goal of continuous-power, wavelength-tunable operation limited only by the physical capabilities of the laser modules.

Future Work Suggestions

- ***Long-Term Feedback Algorithms:*** Adding integral or derivative terms (i.e. full PID control) would enhance rate of convergence as well as restrict steady-state error in the event that simple proportional feedback is not sufficient.

- ***Adaptive Tolerance:*** Adjustable tolerance based upon varying wavelength-dependent noise environments could potentially maximize even further performance.

- ***Expanded Hardware Integration:*** Applying the same architecture to other tunable laser platforms or in integrating automated wavelength-locked etalons could expand the range of applications.

- ***Drift Compensation:*** Long-term monitoring and drift compensation would further enhance stability over long measurement campaigns.

By revealing an entirely automated, Python-controlled solution to powering and wavelength controlling a tunable supercontinuum laser, this work paves the way for even more sophisticated laser instruments in spectroscopy, biomedical imaging, telecommunications, and beyond.

7. References

[1] Dr. Mark Little, "Tunable Lasers Expand for Time-Resolved Spectroscopy" in the Medical Design Briefs, retrieved from https://www.medicaldesignbriefs.com/component/content/article/51699-tunable-lasers-expand-for-time-resolved-spectroscopy
[Accessed: May 10, 2025].

[2] Keyhole Software Team (n.d.) “Scripting with Python and AutoHotkey”, retrieved from https://keyholesoftware.com/scripting-with-python-and-autohotkey [Accessed: May 10, 2025].

[3] Optica Publishing Group (2023) “High-resolution LiDAR for 3D imaging applications”, retrieved from https://opg.optica.org/prj/abstract.cfm?uri=prj-12-8-1709 [Accessed: May 10, 2025].

[4] Ekspla, "Supercontinuum Generation," *Ekspla*. [Online]. retrieved: January 6, 2025 Available: https://ekspla.com/applications/biomedical-scientific/laser-spectroscopy/supercontinuum-generation/

[5] NKT Photonics, “Supercontinuum Generation,” (2019) https://www.nktphotonics.com/products/supercontinuum-white-light-lasers/
[Accessed: May 10, 2025].

[6] R. Paschotta, article on "Tunable Lasers" in the RP Photonics Encyclopedia, retrieved: May 10, 2025 Available: https://doi.org/10.61835/e7x

[7] Nicolai Granzow "Supercontinuum white light lasers: a review on technology and applications", Proc. SPIE 11144, Photonics and Education in Measurement Science 2019, 1114408 (17 September 2019); https://doi.org/10.1117/12.2533094 [Accessed: May 10, 2025].

[8] Findlight Editorial Team. (n.d.). *Supercontinuum Lasers: Discovery and Applications.* Findlight Blog. Retrieved January 6, 2025, from https://www.findlight.net/blog/supercontinuum-lasers-discovery-and-applications/

[9] Thorlabs, *PM100D – Digital Optical Power and Energy Meter Console*. [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=3341. [Accessed: Jan. 6, 2025].

[10] NKT Photonics, *SuperK EXTREME: Supercontinuum Fiber Laser Series*. NKT Photonics A/S. "Figure adapted from NKT Photonics, *SuperK EXTREME: Supercontinuum Fiber Laser Series*, p. 01." [Accessed: May 10, 2025].

[11] NKT Photonics, *SuperK VARIA – Tunable Single-Line Filter*. [Online]. Available: https://www.nktphotonics.com/products/supercontinuum-white-light-lasers/superk-varia/. [Accessed: Jan. 6, 2025].

[12] NKT Photonics, *SuperK CONNECT – High-Precision Fiber Delivery System*. [Online]. Available: https://www.nktphotonics.com/products/supercontinuum-white-light-lasers/superk-connect/. [Accessed: Jan. 6, 2025].

[13] NKT Photonics, *NKT Tools – Python API for SuperK and Varia control.* [Online]. Available: https://nkt-tools.readthedocs.io/en/latest/. [Accessed: Apr. 30, 2025].

[14] P. Cladé, "ThorlabsPM100, version 1.2.2," PyPI, Sep. 1, 2020. [Online]. Available: https://pypi.org/project/ThorlabsPM100. [Accessed: May 10, 2025].

[15] J. Clade, *ThorlabsPM100 – Python driver for Thorlabs PM100D powermeter*. [Online]. Available: https://github.com/clade/ThorlabsPM100 [Accessed: May 10, 2025].

8. Appendix

Full Implementation Script

```
import tkinter as tk
from tkinter import ttk, simpledialog, messagebox
import threading
import time
import csv
import pyvisa
import numpy as np
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
from ThorlabsPM100 import ThorlabsPM100
from nkt_tools.extreme import Extreme
from nkt_tools.varia import Varia
from datetime import datetime

def main():
    # Create the main window
    root = tk.Tk()
    root.title("Laser Power Stabilization System")

    # Create a main frame for padding and layout
    main_frame = ttk.Frame(root, padding="10")
    main_frame.pack(expand=True, fill="both")

    # Add a title label
    title_label = ttk.Label(main_frame, text="Laser Power Stabilization System",
                            font=("Helvetica", 16, "bold"))
    title_label.pack(pady=(0, 10))

    # Create a Treeview to display data
    columns = ("Process", "Wavelength (nm)", "Laser Power Setting (%)", "Measured
Power (µW)")
    tree = ttk.Treeview(main_frame, columns=columns, show="headings", height=15)
    for col in columns:
        tree.heading(col, text=col)
        tree.column(col, stretch=True, anchor="center")
    tree.pack(side="left", fill="both", expand=True)

    # Add a scrollbar
    scrollbar = ttk.Scrollbar(main_frame, orient="vertical", command=tree.yview)
    tree.configure(yscroll=scrollbar.set)
    scrollbar.pack(side="left", fill="y")
    # Status label
    status_label = ttk.Label(root, text="Status: Idle", font=("Helvetica", 12))
    status_label.pack(pady=10)

    # Button frame
    button_frame = ttk.Frame(root, padding="10")
    button_frame.pack()

    # Hardware Setup
    rm = pyvisa.ResourceManager()
    inst = rm.open_resource('USB0::0x1313::0x8078::P0017991::INSTR')
    power_meter = ThorlabsPM100(inst=inst)
    Laser = Extreme()
    Filter = Varia()

    # Ask for target power at startup
    target_power = simpledialog.askfloat("Target Power",
                                         "Enter target power (µW):",
                                         parent=root,
                                         minvalue=1)
    if target_power is None:
        messagebox.showinfo("Info", "No target power entered. Exiting.")
        root.quit()
        return

    root.lift()  # Bring window to front

    # Constants
    MIN_LASER_POWER = 10.0  # Minimum allowed laser power setting (10%) by default
    min_wavelength = 400
    max_wavelength = 840
    NumberOfSteps = 20   # Number of calibration steps

    # Calibration data storage
    calibration_results = []

    def log_entry(step, wavelength, laser_setting, measured_power):
        value = f"{measured_power:.1f}" if measured_power is not None else "N/A"
        item_id = tree.insert("", "end", values=(step, f"{wavelength:.1f}",
f"{laser_setting:.1f}", value))
        tree.see(item_id)  # Auto-scroll to new entry

    def add_separator():
        item_id = tree.insert("", "end", values=("—"*10, "—"*10, "—"*10, "—"*10))
        tree.see(item_id)  # Auto-scroll to separator

    def plot_calibration_curve(cal_data):
        if not cal_data:
            messagebox.showerror("Error", "No calibration data to plot")
            return
        wavelengths = [item[0] for item in cal_data]
        laser_settings = [item[1] for item in cal_data]
        measured_powers = [item[2] for item in cal_data]

        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 10))
        fig.suptitle("Calibration Curve")

        ax1.plot(wavelengths, measured_powers, 'bo-', label="Measured Power")
        ax1.set_xlabel("Wavelength (nm)")
        ax1.set_ylabel("Power (µW)")
        ax1.grid(True)

        ax2.plot(wavelengths, laser_settings, 'ro-', label="Laser Setting")
        ax2.set_xlabel("Wavelength (nm)")
        ax2.set_ylabel("Setting (%)")
        ax2.grid(True)

        plt.tight_layout()
        plt.show()

    def export_data(data, default_name):
        if not data:
            messagebox.showerror("Error", "No data to export")
            return

        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        filename = f"{default_name}_{timestamp}.csv"
        try:
            with open(filename, 'w', newline='') as f:
                writer = csv.writer(f)
                writer.writerow(["Wavelength (nm)", "Laser Setting (%)", "Measured
Power (µW)"])
                writer.writerows(data)
            messagebox.showinfo("Export Successful", f"Data saved to {filename}")
        except Exception as e:
            messagebox.showerror("Export Error", str(e))

    def create_interpolation():
        if not calibration_results:
            messagebox.showerror("Error", "No calibration data available")
            return None
        wavelengths = np.array([x[0] for x in calibration_results])
        settings = np.array([x[1] for x in calibration_results])
        return interp1d(wavelengths, settings, kind='linear',
fill_value="extrapolate")

    # Calibration routine
    def run_calibration():
        try:
            status_label.config(text="Starting calibration...")
            tolerance = 0.5
            max_iterations = 10

            # Initial setup
            Filter.short_setpoint = 495
            Filter.long_setpoint = 505
            Laser.set_power(max(30.0, MIN_LASER_POWER))
            Laser.set_emission(True)
            time.sleep(3)

            # Initial calibration
            current_setting = 30.0
            for _ in range(max_iterations):
                power = power_meter.read * 1e6
                if abs(power - target_power) <= tolerance:
                    break
                current_setting *= target_power / power
                current_setting = max(MIN_LASER_POWER, min(100, current_setting))
                Laser.set_power(current_setting)
                time.sleep(0.5)

            initial_wl = (Filter.short_setpoint + Filter.long_setpoint) / 2
            log_entry("Calibration", initial_wl, current_setting, power)
            calibration_results.append((initial_wl, current_setting, power))
            status_label.config(text=f"Calibrated {initial_wl:.1f}nm: {power:.1f}
µW")
            time.sleep(1)

            # Wavelength sweep
            for step in range(NumberOfSteps):
                new_short = Filter.short_setpoint + 5
                new_long = Filter.long_setpoint + 5
                if (new_short + new_long)/2 > max_wavelength:
                    break

                Filter.short_setpoint = new_short
                Filter.long_setpoint = new_long
                time.sleep(0.5)

                # Power adjustment
                for _ in range(max_iterations):
                    power = power_meter.read * 1e6
                    if abs(power - target_power) <= tolerance:
                        break
                    current_setting *= target_power / power
                    current_setting = max(MIN_LASER_POWER, min(100,
current_setting))
                    Laser.set_power(current_setting)
                    time.sleep(0.5)

                current_wl = (new_short + new_long)/2
                log_entry("Calibration", current_wl, current_setting, power)
                calibration_results.append((current_wl, current_setting, power))
                status_label.config(text=f"Calibrated {current_wl:.1f}nm:
{power:.1f} µW")
                time.sleep(1)
            Laser.set_emission(False)
            status_label.config(text="Calibration complete")
            plot_calibration_curve(calibration_results)
            root.after(0, add_separator)

        except Exception as e:
            Laser.set_emission(False)
            messagebox.showerror("Calibration Error", str(e))
            status_label.config(text="Calibration failed")

    def start_measurement():
        # Get parameters for measurement: start, end, step
        start_wl = simpledialog.askfloat("Start Wavelength", "Enter start (nm):",
                                        parent=root,
                                        minvalue=min_wavelength,
                                        maxvalue=max_wavelength)
        if start_wl is None: return

        root.lift()  # Bring window to front

        end_wl = simpledialog.askfloat("End Wavelength", "Enter end (nm):",
                                      parent=root,
                                      minvalue=start_wl,
                                      maxvalue=max_wavelength)
        if end_wl is None: return

        root.lift()

        step_size = simpledialog.askfloat("Step Size", "Enter step (nm):",
                                         parent=root,
                                         minvalue=1,
                                         maxvalue=end_wl-start_wl)
        if step_size is None: return

        # Ask for laser ON and OFF durations (in seconds)
        on_time = simpledialog.askfloat("Laser ON Time", "Enter laser ON time
(seconds):",
                                        parent=root, minvalue=1)
        if on_time is None: return

        off_time = simpledialog.askfloat("Laser OFF Time", "Enter laser OFF time
(seconds):",
                                         parent=root, minvalue=1)
        if off_time is None: return

        # Start measurement thread with parameters
        threading.Thread(target=lambda: run_measurement(start_wl, end_wl,
step_size, on_time, off_time), daemon=True).start()

    def run_measurement(start_wl, end_wl, step_size, on_time, off_time):
        try:
            if not calibration_results:
                root.after(0, lambda: messagebox.showerror("Error", "Perform
calibration first!"))
                return

            interp_func = create_interpolation()
            calibrated_wls = [x[0] for x in calibration_results]
            min_cal_wl, max_cal_wl = min(calibrated_wls), max(calibrated_wls)

            current_wl = start_wl
            num_steps = int((end_wl - start_wl)/step_size) + 1

            for step_idx in range(num_steps):
                short = round(current_wl - 5)
                long = round(current_wl + 5)
                if not (min_wavelength <= short <= max_wavelength-10):
                    root.after(0, lambda: messagebox.showerror("Error", "Wavelength
out of range!"))
                    break

                # Set filter for current wavelength
                Filter.short_setpoint = short
                Filter.long_setpoint = long
                time.sleep(0.5)
                # Set laser power based on calibration interpolation
                setting = float(interp_func(current_wl))
                setting = max(MIN_LASER_POWER, min(100, setting))
                Laser.set_power(setting)

                # Laser ON phase for specified duration
                Laser.set_emission(True)
                root.after(0, lambda wl=current_wl: status_label.config(
                    text=f"Measuring {wl:.1f}nm - LASER ON for {on_time:.1f} sec"))
                time.sleep(on_time)

                # Laser OFF phase for specified duration
                Laser.set_emission(False)
                root.after(0, lambda wl=current_wl: status_label.config(
                    text=f"Measuring {wl:.1f}nm - LASER OFF for {off_time:.1f}
sec"))
                time.sleep(off_time)

                # Log the measurement (no power meter reading available in open-
loop mode)
                root.after(0, lambda wl=current_wl, set_val=setting:
log_entry("Measurement", wl, set_val, None))
```

```
                if current_wl < min_cal_wl or current_wl > max_cal_wl:
                    root.after(0, lambda: messagebox.showwarning(
                        "Warning", "Extrapolating beyond calibration range!"))

                current_wl += step_size
                current_wl = round(current_wl, 1)

            Laser.set_emission(False)
            root.after(0, lambda: status_label.config(text="Measurement complete"))
            root.after(0, add_separator)

        except Exception as e:
            Laser.set_emission(False)
            root.after(0, lambda: messagebox.showerror("Measurement Error",
str(e)))
            root.after(0, lambda: status_label.config(text="Measurement failed"))

    def start_calibration():
        threading.Thread(target=run_calibration, daemon=True).start()

    # Updated Exit function to turn off the laser and then close the application.
    def exit_application():
        # Cleanup while Tkinter is still alive
        try:
            Laser.set_emission(False)
            inst.close()
            rm.close()

            for item in tree.get_children():
                tree.delete(item)

            if 'title_label' in globals():
                title_label.destroy()
            if 'status_label' in globals():
                status_label.destroy()

            plt.close('all')

        except Exception as e:
            messagebox.showwarning("Warning", f"Cleanup error: {e}")

        # Tkinter shutdown
        try:
            root.quit()
            root.destroy()
        except:
            pass

        # Force cleanup before Python exits
        import gc
        gc.collect()

    # Control buttons
    ttk.Button(button_frame, text="Calibrate",
              command=lambda: threading.Thread(target=run_calibration,
daemon=True).start(),
              width=15).pack(side="left", padx=5)
    ttk.Button(button_frame, text="Measure", command=start_measurement,
              width=15).pack(side="left", padx=5)
    ttk.Button(button_frame, text="Export Calibration",
              command=lambda: export_data(calibration_results, "calibration"),
              width=20).pack(side="left", padx=5)
    ttk.Button(button_frame, text="Plot Calibration",
              command=lambda: plot_calibration_curve(calibration_results),
              width=20).pack(side="left", padx=5)
    ttk.Button(button_frame, text="Exit", command=exit_application,
              width=10).pack(side="right", padx=5)

    root.mainloop()

if __name__ == "__main__":
    main()
```

Online Repository

The complete, up-to-date Python script (and all supporting modules) is also available on GitHub:

https://github.com/alpsalgur/Laser-Power-Stabilization

Please refer to this repository for the latest version, issue tracking, and usage examples.