Development of a thermoelectric temperature controller for photonic microresonators

Trabajo de Fin de Grado

Author:
Diego Menéndez Díaz

Supervised by:
Leopoldo Luis Martín Rodríguez

Universidad de La Laguna
Facultad de Ciencias - sección de Física

Julio 2025

Abstract

This thesis presents the design, implementation, and experimental validation of a digital temperature control system specifically developed for stabilizing photonic microresonators in laboratory environments. The work addresses a critical challenge in photonics research: maintaining precise thermal stability in optical experiments where traditional thermal enclosures cannot be employed due to experimental constraints.

The fundamental motivation stems from the inherent temperature sensitivity of photonic devices, particularly microresonators such as whispering-gallery mode cavities. Temperature variations, whether endogenous (laser-induced heating) or exogenous (ambient fluctuations), significantly affect optical responses through thermal expansion and refractive index changes. These effects induce resonant wavelength shifts that can compromise experimental stability and accuracy. While this sensitivity enables highly sensitive temperature sensors, it presents a major challenge when stable temperature conditions are required for studying nonlinear effects, coupling dynamics, or cavity-enhanced processes.

The developed system employs a closed-loop architecture comprising three essential components: a platinum RTD (PT100) temperature sensor for high-precision measurements, a Peltier thermoelectric module as a bidirectional actuator capable of both heating and cooling, and a sophisticated control algorithm implemented in Python. The system operates without thermal insulation, directly exposed to ambient air, which imposes significant challenges due to convective heat losses and environmental perturbations.

The theoretical foundation encompasses control systems engineering principles, focusing on PID (Proportional-Integral-Derivative) control theory and its digital implementation. The work addresses key challenges in thermal systems including thermal inertia, nonlinear behavior, asymmetric dynamics between heating and cooling modes, and environmental disturbances. The Peltier effect, discovered in 1834, enables bidirectional temperature control through thermoelectric phenomena, where heat absorption or emission occurs at semiconductor junctions when electric current flows.

A critical innovation in this work is the implementation of a dual-mode PID controller with gain scheduling, which addresses the inherent asymmetry between heating and cooling efficiencies in Peltier devices. The controller automatically adjusts PID parameters based on the temperature setpoint, employing different gain values for heating and cooling modes. This adaptive approach ensures optimal performance across the entire operating range (0-50°C) without requiring manual parameter adjustment.

The system architecture integrates a Rigol DP711 programmable power supply for precise voltage control, an Arduino Uno microcontroller for relay-based polarity switching, and an Adafruit MAX31865 RTD amplifier for high-precision temperature measurements. The control software, developed in Python, implements real-time data acquisition, PID algorithm execution, and user interface functionality through a graphical interface.

Experimental results demonstrate the system's effectiveness under challenging open-air conditions. Without thermal insulation, the controller achieves temperature stability within ± 0.2°C peak-to-peak fluctuations over 5-minute intervals. The implementation of an empirical offset correction function effectively eliminates steady-state errors, reducing temperature offsets from up to 2.2°C to negligible levels. When basic thermal insulation is applied, stability improves to approximately ± 0.06°C, approaching the performance of commercial temperature controllers.

The gain scheduling strategy successfully addresses the asymmetric behavior of Peltier devices, with proportional gains of 0.75 for cooling and 0.65 for heating modes. Integral and derivative gains are dynamically adjusted based on temperature setpoints, with higher values required for extreme temperatures and minimal gains near room temperature. The system maintains stability across the full operating range while preventing oscillations and overshoot.

Key achievements include the development of a robust temperature control system that operates effectively in open-air laboratory environments, the implementation of adaptive PID control with gain scheduling, and the creation of a user-friendly interface for real-time monitoring and control. The system demonstrates remarkable stability given the experimental constraints, achieving performance levels suitable for photonic research applications.

The work contributes to the field by providing a practical solution for temperature control in photonic experiments where traditional thermal enclosures are not feasible. The dual-mode PID approach with gain scheduling represents a novel adaptation of control theory to address the specific challenges of thermoelectric temperature control. The empirical offset correction methodology offers a simple yet effective approach to eliminating steady-state errors in practical applications.

Future work could explore the integration of advanced control algorithms such as model predictive control, the development of compatible thermal insulation solutions, and the extension of the system to multi-zone temperature control for complex photonic devices. The modular design and open-source software implementation provide a foundation for further development and adaptation to specific experimental requirements.

This thesis demonstrates that precise temperature control in challenging laboratory environments is achievable through careful system design, adaptive control strategies, and empirical optimization. The developed system provides a valuable tool for photonics research, enabling stable experimental conditions essential for advancing our understanding of optical phenomena and developing next-generation photonic devices.

Resumen

Este trabajo presenta el diseño, la implementación y la validación experimental de un sistema digital de control de temperatura desarrollado específicamente para la estabilización de microresonadores fotónicos en entornos de laboratorio. El trabajo aborda un desafío crítico en la investigación en fotónica: mantener una precisión térmica en experimentos ópticos donde no es posible emplear recintos térmicos tradicionales debido a las restricciones experimentales.

La motivación fundamental surge de la sensibilidad intrínseca de los dispositivos fotónicos a la temperatura, en particular de los microresonadores como las cavidades de modo susurrante ("whispering-gallery mode"). Las variaciones de temperatura, ya sean endógenas (calentamiento inducido por láser) o exógenas (fluctuaciones ambientales), afectan significativamente las respuestas ópticas mediante la expansión térmica y los cambios en el índice de refracción. Estos efectos inducen desplazamientos en la longitud de onda de resonancia que pueden comprometer la estabilidad y la precisión experimental. Si bien esta sensibilidad permite dispositivos sensores de temperatura de alta resolución, representa un gran desafío cuando se requieren condiciones térmicas estables para estudiar efectos no lineales, dinámicas de acoplamiento o procesos potenciados por cavidades.

El sistema desarrollado emplea una arquitectura de lazo cerrado que comprende tres componentes esenciales: un sensor de temperatura de platino (RTD PT100) para mediciones de alta precisión, un módulo termoeléctrico Peltier como actuator bidireccional capaz de calentar y enfriar, y un sofisticado algoritmo de control implementado en Python. El sistema opera sin aislamiento térmico, expuesto directamente al aire ambiente, lo que impone desafíos significativos debido a las pérdidas por convección y las perturbaciones ambientales.

La base teórica abarca principios de ingeniería de sistemas de control, centrándose en la teoría de control PID (proporcional-integral-derivativo) y su implementación digital. El trabajo aborda retos clave en sistemas térmicos, incluyendo la inercia térmica, el comportamiento no lineal, la dinámica asimétrica entre modos de calentamiento y enfriamiento, y las perturbaciones ambientales. El efecto Peltier, descubierto en 1834, permite el control bidireccional de la temperatura mediante fenómenos termoeléctricos, donde se absorbe o emite calor en las uniones semiconductoras al circular corriente eléctrica.

Una innovación crucial en este trabajo es la implementación de un controlador PID de doble modo con programación de ganancias ("gain scheduling"), que enfrenta la asimetría inherente entre las eficiencias de calentamiento y enfriamiento en dispositivos Peltier. El controlador ajusta automáticamente los parámetros PID según el punto de consigna de temperatura, empleando valores de ganancia diferentes para

los modos de calentamiento y enfriamiento. Este enfoque adaptativo garantiza un rendimiento óptimo en todo el rango de operación (0–50°C) sin necesidad de ajuste manual de parámetros.

La arquitectura del sistema integra una fuente de alimentación programable Rigol DP711 para control de voltaje preciso, un microcontrolador Arduino Uno para el conmutado de polaridad mediante relés, y un amplificador de RTD Adafruit MAX31865 para mediciones de temperatura de alta precisión. El software de control, desarrollado en Python, implementa adquisición de datos en tiempo real, ejecución del algoritmo PID y funcionalidad de interfaz gráfica de usuario.

Los resultados experimentales demuestran la efectividad del sistema en condiciones desafiantes al aire libre. Sin aislamiento térmico, el controlador logra una estabilidad de temperatura dentro de ±0,2°C de fluctuación pico a pico en intervalos de 5minutos. La implementación de una función empírica de corrección de offset elimina eficazmente los errores en estado estacionario, reduciendo los desfases de hasta 2,2°C a niveles despreciables. Al aplicar un aislamiento térmico básico, la estabilidad mejora a aproximadamente ±0,06°C, acercándose al rendimiento de controladores comerciales.

La estrategia de programación de ganancias aborda con éxito el comportamiento asimétrico de los dispositivos Peltier, con ganancias proporcionales de 0,75 para enfriamiento y 0,65 para calentamiento. Las ganancias integral y derivativa se ajustan dinámicamente según el punto de consigna, requiriendo valores más altos en temperaturas extremas y valores mínimos cerca de la temperatura ambiente. El sistema mantiene la estabilidad en todo el rango de operación, previniendo oscilaciones y sobreimpulsos.

Los logros clave incluyen el desarrollo de un sistema de control de temperatura robusto que opera eficazmente en entornos de laboratorio al aire libre, la implementación de control PID adaptativo con programación de ganancias y la creación de una interfaz de usuario amigable para la monitorización y control en tiempo real. El sistema demuestra una estabilidad notable dadas las restricciones experimentales, alcanzando niveles de rendimiento adecuados para aplicaciones de investigación en fotónica.

El trabajo contribuye al campo al ofrecer una solución práctica para el control de temperatura en experimentos fotónicos donde no es factible utilizar recintos térmicos tradicionales. El enfoque de PID de doble modo con programación de ganancias representa una adaptación novedosa de la teoría de control para abordar los desafíos específicos del control termoeléctrico de temperatura. La metodología empírica de corrección de offset ofrece un enfoque simple y eficaz para eliminar errores en estado estacionario en aplicaciones prácticas.

El trabajo futuro podría explorar la integración de algoritmos de control avanzados como el control predictivo basado en modelo (MPC), el desarrollo de soluciones de aislamiento térmico compatibles y la extensión del sistema al control de temperatura de múltiples zonas para dispositivos fotónicos complejos. El diseño modular y la implementación de software de código abierto proporcionan una base para su desarrollo y adaptación a requisitos experimentales específicos.

Esta tesis demuestra que el control preciso de la temperatura en entornos de laboratorio desafiantes es alcanzable mediante un diseño cuidadoso del sistema, estrate-

gias de control adaptativo y optimización empírica. El sistema desarrollado ofrece una herramienta valiosa para la investigación en fotónica, permitiendo condiciones experimentales estables esenciales para avanzar en la comprensión de fenómenos ópticos y en el desarrollo de dispositivos fotónicos de próxima generación.

Contents

Chapter 1

Introduction

El control preciso de la temperatura es esencial en fotónica, donde pequeños cambios térmicos pueden alterar resonancias ópticas y comprometer los resultados. Este trabajo presenta un controlador digital de temperatura para microresonadores fotónicos en aire libre, basado en un PID dual y programación de ganancias, capaz de operar en entornos de laboratorio sin aislamiento.

Precise temperature control is vital in photonics, where tiny thermal drifts can shift optical resonances and undermine experimental accuracy. This thesis develops a digital, open-air temperature controller - with dual-mode PID and gain scheduling - to stabilize photonic microresonators under challenging laboratory conditions.

1.1 Context and Relevance

Temperature is a fundamental parameter in any physical experiment. Due to the thermodynamic nature of matter, heat is inevitably generated in virtually all physical processes. For instance, laser diodes heat up due to Joule effect, laser crystals accumulate energy through optical absorption, and even mechanical components suffer heating through friction. In many laboratory contexts, thermal variations are either neglected or assumed to be negligible, which can lead to inaccurate results when this assumption is not valid.

In photonics, temperature plays a particularly critical role. Variations in temperature, whether endogenous (e.g., laser-induced heating) or exogenous (e.g., ambient fluctuations), can significantly affect the optical response of a system through two main effects: thermal expansion of the material, quantified by the thermal expansion coefficient α , and changes in the refractive index with temperature, quantified by the thermo-optic coefficient β. In optical resonators, both effects induce shifts in the resonant wavelengths of the cavity. While this sensitivity has been exploited to design highly sensitive temperature sensors based on whispering-gallery mode resonators [1], it represents a major challenge in experiments where stable temperature conditions are required.

In particular, experiments with integrated photonic microresonators, such as microdisks, demand precise thermal control. Small temperature drifts can hinder the ability to fix the resonance of the cavity at a desired wavelength, compromising the

stability needed to study nonlinear effects, coupling dynamics, or cavity-enhanced processes.

1.2 Purpose of this work

This project presents the design and implementation of a digital temperature controller aimed at stabilizing the temperature of on-chip photonic microcavities. The system has been conceived for operation in a laboratory environment where no thermal enclosure can be applied directly to the sample-sensor-Peltier interface, as required by the optical setup. This imposes strong constraints due to convective heat losses and ambient fluctuations. To overcome these limitations, the controller integrates a dual-mode PID algorithm implemented in Python, automatic gain adjustment (gain scheduling), and an empirical correction strategy to compensate for the steady-state error.

The control strategy is implemented through a closed-loop system involving a Peltier module, a platinum RTD sensor, and a programmable power supply. The overall architecture has been designed to offer stability, flexibility, and compatibility with photonic experimental setups, making it suitable for the precise thermal regulation required in advanced optics research.

Chapter 2

Theoretical background

Este capítulo proporciona los fundamentos teóricos necesarios para llevar a cabo este trabajo. Se muestran los principios generales de los controladores de temperatura, y en concreto de los controladores PID (proporcional-integral-derivativo), aplicable en todo tipo de sistemas de control. Se introduce también a la física detrás del resto de elementos utilizados, como las placas Peltier y los sensores de temperatura, ambos basados en principios de la misma naturaleza: el efecto termoeléctico.

In order to establish the bases of knowledge required to develop this work, this chapter was created. This includes the general principles of temperature control, from control systems engineering to PID (proportional-integral-derivative) controllers. Also includes a brief explanation of the thermoelectric phenomena behind Peltier plates and temperature sensors.

2.1 General principles of temperature control

Control systems engineering is a fundamental discipline, concerned with influencing the behavior of dynamic systems to achieve desired performance [2]. They can be classified into two main categories: *open-loop* and *closed-loop* systems. In open-loop systems, the output has no effect on the control action, and it can be useful in cases where there is a well-known response. In a closed-loop or feedback system, the output directly affects the control action, and it allows precise regulation to face variations in operating conditions and external perturbations.

This feedback architecture is usually employed in temperature controllers, and is particularly useful when a constant temperature is desired, since non-linear and delayed environmental perturbations strongly affect the system. They consist of three main components:

- A **sensor**, to measure temperature
- An **actuator**, to heat or cool
- A **control algorithm**, to set an appropriate response

The sensor measures the temperature at every instant, the control algorithm sets a response depending on it, and the actuator brings the temperature to a predefined

target temperature - called the *setpoint*. Several types of sensors can be used such as regular thermometers, thermistors, thermocouples and resistance temperature detectors (RTDs) [3]. Actuators may include all kind of resistive heating elements, fans, fluid refrigeration systems, or thermoelectric modules such as Peltier plates which can both heat and cool.

2.1.1 Physical challenges in thermal systems

In general, there are some challenges to face in the practice of temperature controllers:

- **Thermal inertia and resistance**: The change of temperature of matter is not instantaneous. It means that after a heat input the temperature of a body will change after a delay, which is determined by the heat capacity of the system [4]. Also is important to isolate hot and cold parts of the system, to avoid heat transmission by conduction that could affect the temperature of the sample.

- **Nonlinearity**: In most common actuators the relationship between the control input (e.g., voltage in Peltier plates) and the temperature change induced is nonlinear [5], due to the saturation effects, thermoelectric efficiency variations, dependence of thermal conductivity with temperature, etc. This creates the need for a model that responds adaptively to the measurements instead of finding an analytical model based strategy.

- **Asymmetric dynamics**: Frequently, actuators such as Peltier modules, which can both heat and cool, present an asymmetric response. The difference between the behavior in both modes is huge and aims for completely different tuned controllers. It is caused by the big efficiency of heating compared to cooling.

- **Environmental perturbations**: Any external disturbance can affect the temperature of the sample/sensor, due to convective losses [6], and ambient temperature fluctuations, which reinforces the need for an adaptive closed-loop strategy.

These properties should be considered to achieve a desired temperature. When putting the controller into practice, there are also some specific experimental constraints to take into account to perform adequate thermal regulation. These considerations directly affect the response of the system. The most relevant ones are:

- The sensor, actuator and the sample must be **well thermally coupled**. To avoid excessive delays and misrepresentations of the temperature, they should be put in direct contact, maximizing the heat transfer.

- The **actuator range** of operation must be reasonably selected to get a stable and predictable response, and to avoid saturation/overheating.

- To maximize the cooling capacity of the controller there must be ensured enough **heat dissipation** efficiency. It is widely recommended to use fans and heatsinks, to help heat removal and therefore increase cooling performance.

- The control loop should operate in an appropriate sampling rate, to be updated for any temperature deviations. Also the actuator should have sufficiently high resolution to create smooth responses.

2.1.2 Relevance in this work

In this work, thermal resistance between the module components causes a slow dynamic of the system. The use of a heatsink and a cooling fan is essential for good results. Also, heat transfer between the components is affected by conduction.

This system in particular counts with a Peltier plate as an actuator, a platinum resistance as a sensor, and a PID control algorithm built in a Python script - more details in the sections below.

2.2 Thermoelectric modules and sensors

Both thermoelectric modules and temperature sensors are used in this project, and they work on similar principles - they have a relation between current and temperature. Thermoelectric modules like Peltier plates work on the Peltier effect, which allows to create a difference of temperature between the sides of the plate when a current flows. Platinum resistances work as temperature sensors since they have a resistance which depends on the temperature.

2.2.1 Peltier plates

The Peltier effect, discovered in 1834, is a thermoelectric phenomenon which consists of the absorption or emission of heat at the junction between two different conductors or semiconductors when an electric current flows through them. At the atomic level, this occurs because the electrons traveling from one material to the other gain or lose thermal energy depending on the electronic structure of the components and the current's direction [7]. Unlike the Joule effect, which only causes heat emission, the Peltier effect allows to heat or cool.

The quantity of heat Q transferred by the Peltier effect at the junction of the two different materials can be expressed as:

$$Q = \Pi \, I \, t \tag{2.1}$$

where Π is the Peltier coefficient of the junction in volts (V), $I = \frac{V}{R}$ is the intensity of the current in amperes (A), and t is the time. This Q can be positive or negative depending on the direction of the current I.

Peltier plates use the Peltier effect, so when a voltage is applied between its terminals it creates a difference of temperature between the faces of the plate, absorbing heat from the cold side and releasing it on the hot side [8]. These devices are a kind of heat pumps, which are made of multiple thermocouples composed of n-type and

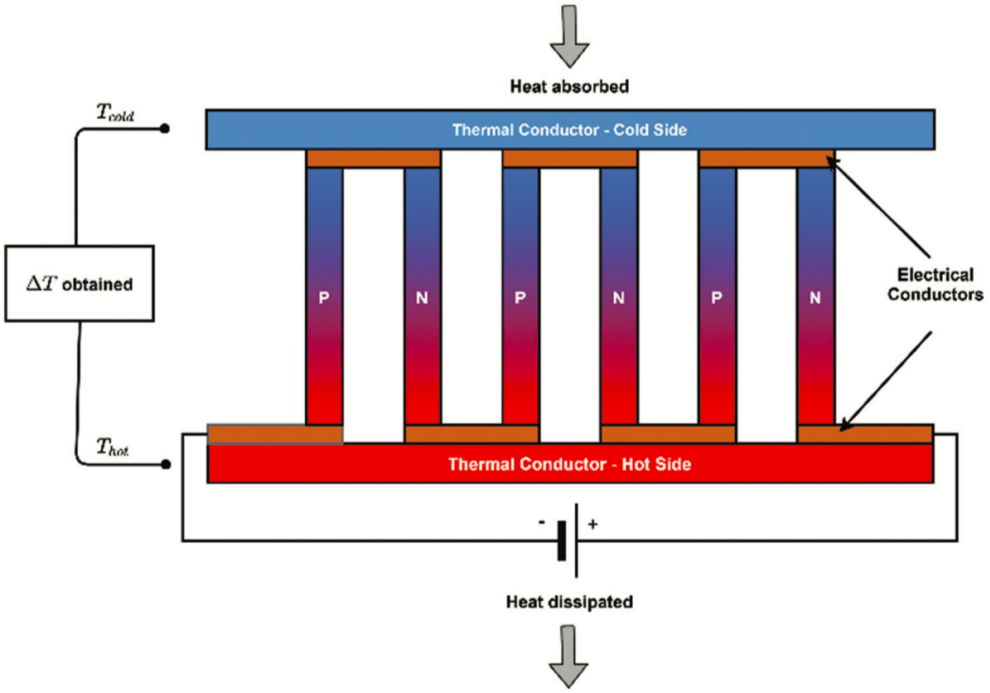

Figure 2.1: Schematic of a thermoelectric cooler. Source:[8]

p-type semiconductors connected in series (but thermally connected in parallel), as shown in Figure 2.1. Then these thermocouples are sandwiched between two ceramic plates for electrical insulation and protection. The charge carriers - electrons and holes - transport heat between sides, allowing the module to actively cool on one face and heat on the opposite. When the current is inverted ($I < 0$), the heat flow is inverted too ($Q < 0$), exchanging the cold and hot sides.

To heat, they present high efficiency because the Peltier effect adds to the Joule effect: in addition to working as a heat pump the module itself gets affected by Joule effect's energy dissipation.

However, as a refrigerator, Peltier plates typically exhibit low efficiency, compared to traditional refrigeration systems. In the practice, their behavior is nonlinear and very sensitive to external disturbances, so the temperature may not be stable and change depending on factors like room temperature, time functioning, etc. Also thermal hysteresis will affect the module as a refrigerator when changing cycles of cooling and heating. To maximize the cooling performance, it is necessary to ensure efficient heat dissipation on the other face. Metallic heat-sinks are typically attached to the hot side and often combined with forced convection systems like fans. Nevertheless, their compactness and ability to precisely control localized temperatures make them ideal for electronic cooling, thermal stabilization of sensors, and portable refrigeration.

2.2.2 RTD temperature sensors

Resistance temperature detectors (RTD) are sensors that operate following the principle of variable resistance of some materials with temperature [9]. In general, any

metal would increase its electrical resistance with temperature. Among all the types of RTDs, plantinum-based sensors (PT100) are the most commonly used for its accuracy, linearity, stability, etc. These kind of sensors are widely used in industrial and scientific applications which require precise temperature monitoring.

In PT100 sensors is precisely 100Ω at $0°C$ and it increases with temperature, approximately following the Callendar - Van Dusen equation [10]:

$$R(T) = R_0 \left(1 + AT + BT^2 + C(T - 100)T^3\right) \tag{2.2}$$

where R_0 is the resistance at $0°C$, and A, B, and C are empirically determined constants. The cubic term is often negligible for temperatures above $0°C$.

2.3 PID control theory

The **proportional-integral-derivative** (PID) controller is one of the most widely used feedback control systems in the industry [11], and so in temperature controllers. Since the non-linear behavior of Peltier plates and the simplicity of its implementation, PID is chosen to be the mechanism of control.

The time-dependent output of a PID controller can be written as:

$$output(t) = K_p e(t) + K_i \int dt\, e(t) + K_d \frac{d}{dt} e(t) \tag{2.3}$$

where $e(t)$ is the error (the difference between the setpoint and the measured value) at a time t, and the constants K_α are the proportional, integral and derivative gains, respectively [12].

The exact value of these gains will depend on the intrinsic characteristics of the system, and they must be selected by using the methods described in the next section, a process known as tuning. Each term of 2.3 has its specific function in the overall behavior of the controller.

Proportional term

Main term, usually driven by the greater gain, and it produces an output proportional to the error at a given time $t = t_o$:

$$P_{out}(t_o) = K_p\, e(t_o) \tag{2.4}$$

If the proportional gain K_p is too high, the system may become unstable. If it is too low, the response may be less sensitive to variations in the error $e(t)$ [13].

Integral term

The integral term produces an output proportional to the integral of the error:

$$I_{out}(t_o) = K_i \int_0^{t_o} e(t)dt \tag{2.5}$$

This considers the cumulative error for a period of time, allowing to reduce long-term and steady-state errors [11]. The integral acts as a low-frequency filter, so this

kind of errors are targeted and progressively reduced to zero. A high integral gain or big intervals of integration may cause overshoot or wind-up.

Derivative term

Proportional to the first derivative of the error:

$$D_{out}(t_o) = K_d \left. \frac{d\,e(t)}{dt} \right|_{t=t_o} \tag{2.6}$$

This term considers the rate of change of the error. The derivative acts as a high-frequency filter, so it reduces the overshoot and improves the transient response. However, K_d is usually the smallest of the constants, and is even considered zero in many cases [11].

2.3.1 Tuning

The main task of implementing a PID control is the tuning of the K_α parameters, which is not trivial [14]. There are several theoretically derived methods to determine their optimal value such as Ziegler-Nichols methods or Cohen-Coon Method, which provide tuning formulas based on experiments, or more modern techniques such as auto-tuning, genetic algorithms, etc. But in practice, it has been found that it is commonly better to use simpler procedures such as empirical trial-and-error methods [11][12]. Figure 2.2 schematically illustrates different types of step responses for various parameter configurations, highlighting the trade-offs between stability and response speed.

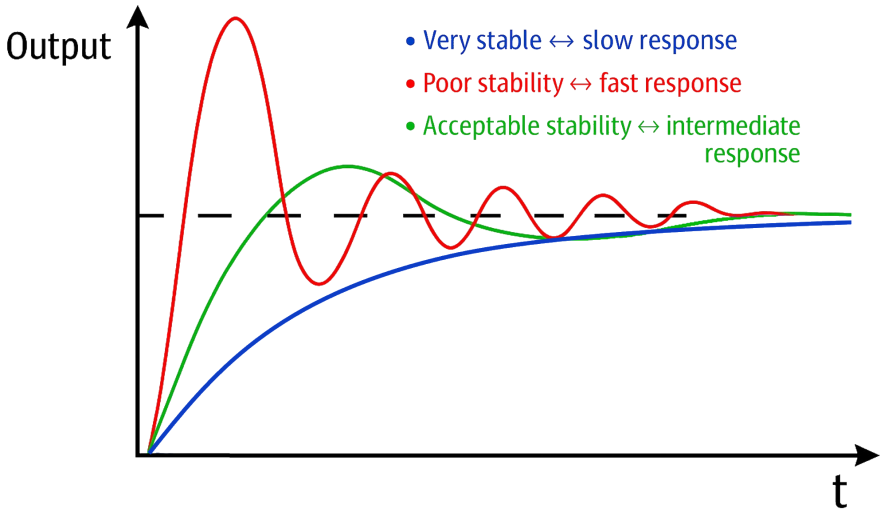

Figure 2.2: Qualitative and schematic representation of the step response for different control parameter settings. The dashed horizontal line represents the setpoint of a generic control system. The green curve reflects an optimized compromise, outperforming the overdamped (blue) and underdamped (red) cases.

In this work in particular, due to the experimental nature of the project, the lack of precision in the active components, and the complexity of the system model, which is unknown, the tuning has been made through a trial-and-error approach.

This method consists of consecutive trials following some rules, adjusting the gains to obtain a rapid and stable response, minimize overshoots, and eliminate or reduce steady-state error [15]. Based on the bibliography, the steps of the procedure are the following:

1. Proportional gain tuning

- Set $K_i = K_d = 0$.
- Gradually increase K_p to get a faster response.
- Stop before the oscillations start.

It is important to consider here that the output is going to be directly proportional to the error - if $K_p = 1$ then the output value is equal to the error. The value of K_p will highly depend on the characteristics of the system.

Once this is done the value of K_p should be optimally tuned. However, a steady-state error (offset) might remain.

2. Integral gain tuning

- Increase K_i from zero to a small value (e.g., $K_i = 0.1\,K_p$).
- The offset will decrease, but overshoot or over-oscillations may appear.

Choose the value which minimizes offset but lets the oscillations start. In some cases, this could be the last step, resulting in a PI controller, where the K_p and K_i gains already satisfy the objective of the controller.

3. Derivative gain tuning

- Now slightly increase K_d (e.g., $K_d = 0.01\,K_p$).
- Repeat iteratively, until the overshoot disappears.

In some cases users have found that adding the K_d produces more oscillations and instability [11], [13]. In these cases K_d should be as small as possible, or even reduced to zero.

4. Fine tuning and validation

- Readjust the parameters through small variations to optimize the trade-offs between response time, overshoot and robustness [16].
- Apply different setpoints to validate the performance of the system.

2.3.2 Digital implementation

Most PID controllers in the industry are implemented digitally, using microprocessors or microcontrollers [17]. This task requires a time-discrete version of the equation of PID 2.3, so that it can be used in a code. Integral and derivative cannot be implemented in the code as they are - they work on continuous time intervals. Therefore, a summatory and a difference appear to replace them. The discrete form of the PID equation can be written as:

$$output[k] = K_p\, e[k] + K_i \sum_j e[j] + K_d\, (e[k] - e[k-1]) \tag{2.7}$$

where $output[k]$ and $e[k]$ are the output of the PID algorithm and the error at the instant k, $e[j]$ represents the past errors up to instant k, and $e[k-1]$ is the error at a previous instant. Therefore, the summatory $\sum_j$ is replacing the integral and the difference $e[k] - e[k-1]$ acts as the derivative.

This implementation may present some challenges and considerations.

Integral windup effect

The integral of the error is the area under the curve, but it can be represented as the sum of previous errors. The main issue here is to properly select the integral limits - considering all the errors from the beginning could cause overload or unnecessary effects.

The integral windup effect occurs when the output of the controller saturates while there is a significant error. In this situation, when the signal is still too far from the setpoint, the integral term accumulates these values, leading to excessive response, even causing the integral term to become larger than the proportional term.

This can be mitigated by several anti-windup techniques ([18], [19]). In this work, the windup is avoided by choosing a relatively small interval of integration, i.e. a few previous error terms in the summatory, instead of considering all.

Sampling period

The sampling period, understood as the number of data per unit of time, or the distance between k and $(k-1)$ instants in the equation 2.7, must be selected correctly for a well-performed controller.

A too long period will produce delays in the behavior, while too short period could produce numerical instability and increase the computational load .

2.3.3 Special considerations for temperature control

A temperature controller uses the equation 2.3 with the error defined as:

$$e(t) \equiv |T_{\text{measured}}(t) - \tau| \tag{2.8}$$

where T_{measured} is the time-dependent value measured by the feedback element (sensor) and τ is the temperature setpoint selected by the user.

For a temperature controller system, some extra considerations must be taken. [20] In general, the active components which allow to rise or lower the temperature are not the same. Thus the PID control heater may not be the same as the cooler. This occurs because the characteristics of the systems when heating or cooling might be physically different.

In the context of this work, the Peltier plate is the active component which does both heat and cool, depending on the circumstances. Heating is considerably efficient, while cooling depends on natural dissipation, presenting much more difficulties. Therefore, considering that they are described by a different nature, they will need an asymmetric PID which differentiates whether the system is cooling or heating.

It is also important to consider thermal inertia, thermal hysteresis, external disturbances and conditions like the room temperature.

Chapter 3

System design

Este capítulo muestra cómo se ha diseñado el sistema, desde los componentes físicos hasta el diseño del software. Se explican los componentes utilizados: placa Peltier, fuente de alimentacion DC programable, sensor de temperatura tipo RTD, un relé para invertir la polaridad de la placa, y un arduino; y de qué forma están integrados. Además se explica el flujo de información y el bucle de control principal del sistema, que se ha desarrollado en un script de Python.

The system was designed in a particular way, including hardware and software architecture. The components are shown in the sections below, as well as the connection between them. Finally, the control pipeline of the system and the implementation in Python is explained.

3.1 Experimental setup

The temperature control system was designed with careful consideration of component selection and integration to achieve a precise thermal regulation.

The initially proposed components to make the temperature controller were a Peltier plate, as the principal active thermal element, a programmable DC power supply controlled by a computer, and a temperature sensor. To achieve the objective also were necessary an Arduino microcontroller, a relay for switching polarity, and a heatsink with a cooling fan to enhance thermal efficiency.

The complete experimental setup is summarized in Figure 3.1. Also, the real setup with the real components is shown in Figure 3.2.

3.1.1 Thermoelectric module (Peltier plate)

The primary thermal control component is the **TEC-12706** Peltier thermoelectric module [21], selected for its balance between cost-effectiveness and performance, its size (40 x 40 x 3.9 mm) and versatility. It is widely used in this kind of applications [22], [23].

It operates up to 12 V with a maximum intensity of 6A, and its made of bismuth tin (BiSn) and covered by two ceramic plates of Alumina (Al_2O_3). It has two terminals which are connected to the power supply.

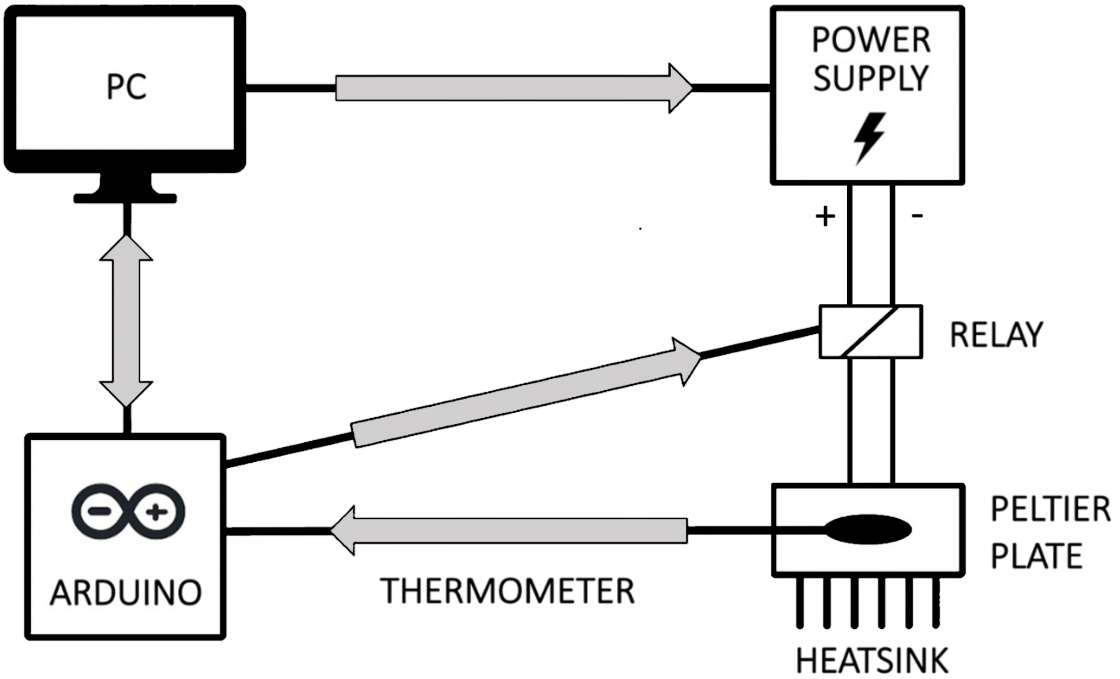

Figure 3.1: Schematic of the hardware architecture for the temperature controller. The gray arrows represent the control pipeline of the system.

For this controller, the input range supplied to the Peltier is 0 - 7 V, to avoid overheating, satisfying enough the specifications of the project.

3.1.2 Power supply (Rigol DP711)

The DC power supply **Rigol DP711** [24] (*Rigol Technologies, Inc.*) is used in the experiment to provide the current to the Peltier module. It has an output range of 0 - 30 V and up to 5 A, and a precise and programmable power control, aligning well with the specifications of the Peltier. The maximum voltage applied will be 7V.

The DP711 is powered by the standard AC mains outlet (220 V), and it is connected to the computer by a serial port. For remote control in automatized systems, it includes a standard RS232 interface, enabling communication via SCPI commands (Standard Commands for Programmable Instruments) [25].

Direct analog voltage control is implemented instead of pulse-width modulation (PWM), to avoid stress on the Peltier p-n junctions. Rapid on-off cycling can cause excessive thermal and mechanical stress, potentially reducing the device lifespan [26].

3.1.3 Microcontroller (Arduino UNO)

To achieve the objective an **Arduino UNO R3** microcontroller [27] was employed, which was involved in different tasks. It serves as the interface between temperature sensor, relay control and computer, which runs the main control software. The Arduino was selected for its analog input and output capabilities, sufficient analog and digital I/O pins and its widespread compatibility.

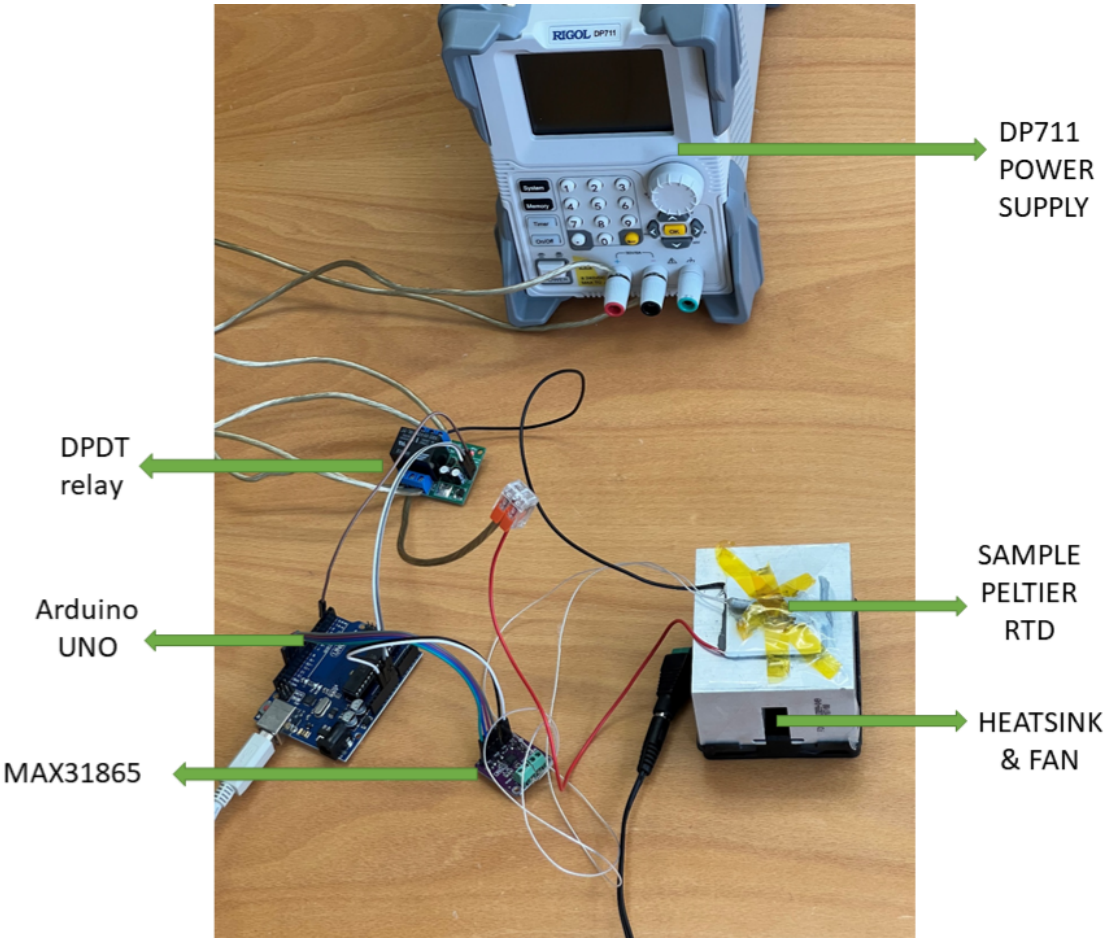

Figure 3.2: Real setup and hardware components of the project.

The microcontroller has the following missions:

- Read and digitalize the RTD temperature measurements
- Control the relay functioning
- Supply power (5 V) to the relay module
- Send the data to the computer via serial

Temperature sensing (RTD)

The measurement of the temperature is carried out by the RTD PT100 sensor, which is connected to the **Adafruit PT100 RTD Temperature Sensor Amplifier - MAX31865** [28]. The PT100 was selected due to its high accuracy and stability over a wide temperature range. The MAX31865 breakout board handles the signal conditioning and analog-to-digital conversion, required to interface the analog sensor with the Arduino UNO via Serial Peripheral Interface (SPI) communication.

Polarity control (DPDT Relay)

The power supply DP711 has a positive range of output voltages, so the changing of polarity of the Peltier plate should be performed by an additional element. A double-pole-double-throw (DPDT) relay - model **DR25E01** satisfies this requirement, switching the sign of the intensity that the DP711 supplies to the Peltier. The module is powered (5 V) and controlled by the Arduino, with a digital output pin configured to trigger the relay by a short pulse. The trigger can be activated manually, but is preferred digitally. This element allows toggling between heating and cooling modes by sending a specific command.

The code uploaded to the Arduino which manages temperature reading and polarity control is `arduino.ino` [1], available online. It includes the calculations for the PT100 calibration and the sampling period, and also the relay switching strategy by sending trigger pulses.

3.1.4 Thermal management and integration

In order to optimize the cooling mode and reach lower temperatures, a finned aluminum heatsink (80 x 80 x 30 mm) and a cooling fan (80 x 80 mm) were attached to the opposite face of the Peltier plate. The fan requires an input of 12 V which was supplied by an ordinary charger connected to the mains outlet.

In the practice, one of the faces of the Peltier plate was fixed as the reference, and the heatsink device placed on the opposite one, but hot and cold sides might change - the system alternates between heating and cooling modes as needed. When cooling, the efficiency is enhanced due to the heat dissipation on the other face. When heating, the heatsink is on the cold side, so it is useless. This should not be a problem because the performance that is needed to enhance is the cooling one, and heating is much more efficient, especially in the range of wanted temperatures.

[1] https://github.com/menendezdiaz/TemperatureController.git

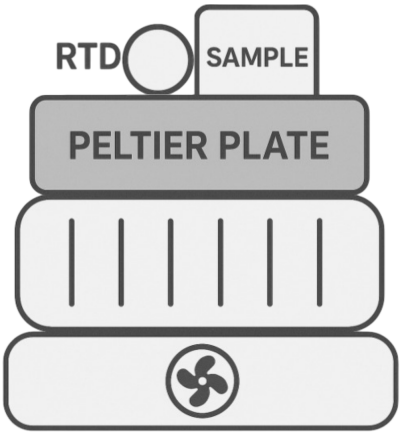

Figure 3.3: Schematic of the integration of the sample. It should be placed in contact with the RTD sensor and both in the same conditions to get affected by the Peltier plate, and the other side of the Peltier is attached to the heatsink and the fan

All thermal interfaces between the heatsink, the Peltier, the RTD and the sample were bonded together using thermal compound, to optimize heat transfer and reduce thermal resistance.

The measurements were conducted on one side of the Peltier module, designated as the test face - the surface of interest. Therefore, the RTD must be placed **in contact** with the surface of interest and **next to** the sample (as shown in Figure 3.3) , and **thermally insulated** if possible, to minimize delays and external disturbances.

Additionally, the operating parameters of the components described above are summarized in the Table 3.1.

Table 3.1: Operating parameters of the main components used in the experimental setup.

Component	**Operating Conditions**
Peltier module	-7 – 7 V DC supply, max 5 A
Power supply (DP711)	Output voltage: 0 – 7 V; Current limit: 5 A
RTD temperature sensor	Operating range: -50 – 150°C
Arduino board	5 V logic level (USB)
Relay module	5 V (Arduino pin)
Cooling fan	12 V DC supply voltage
Heatsink	Passive aluminum heatsink, fan-assisted
Temperature control range	0 – 50°C
Sampling interval	2.5 samples per second

3.2 Control pipeline and communication

Once the hardware components were installed, they must be connected through a specific communication pipeline. The central logic is implemented in Python

(detailed in the next section 3.3), executed in a personal computer (PC), and it controls all the communication between components.

Figure 3.1 also gives some hints about the control pipeline: the gray arrows represent the flux of information. This project aims to control the temperature of the sample using a closed-loop mechanism. In other words, its objective is to minimize the error between the temperature **read by the sensor** and the temperature **set by the user** (setpoint).

3.2.1 Data flow and system timing

The control pipeline begins at the sample, whose temperature is measured by the RTD and sent to the Arduino. This Arduino reports the most recent, already calibrated, temperature reading to the PC at each sampling interval - 0.4 seconds on average. Then the PC, running the control algorithm in Python, processes this value, and using the PID calculations, it determines the output voltage needed to reach the setpoint. Through SCPI commands, the PC sets the voltage with which the DP711 supplies to the Peltier plate, resulting in a decrease or an increase in the temperature of the sample. However, this temperature can be above or below the setpoint, so the PC tells the DPDT relay through the Arduino whether to heat or cool, only by inverting the voltage supplied to the Peltier plate.

This modular and sequential design ensures low latency responses, precise regulation, and a straightforward integration of the control architecture.

3.2.2 Communication channels

The flow of information in the loop is organized in different and independent communication channels between the components. Each channel depends on the specifications of each component, as shown below. The most remarkable communication channels in this system are summarized in Table 3.2.

Arduino - PC: Connected through a USB-to-serial adapter. The flow of information here is bidirectional, because the Arduino sends the temperature data to the PC, and the PC sends specific commands to the Arduino to control the relay. The Arduino contains a code uploaded, which is read by the PC using the `pySerial` library [29].

Arduino - RTD sensor: The RTD, which has a temperature-dependent resistance, is connected to the Arduino using the MAX31865 amplifier. The MAX31865 supplies an excitation current to the RTD, measures the voltage drop to obtain the small resistance changes, and performs digital conversion through the SPI protocol. This allows a high-precision temperature measurement.

Arduino - DPDT relay: The three connections of the DPDT relay join the Arduino to (1) 5 V, (2) ground, and (3) a digital pin. Through this pin, the Arduino sends a short pulse ($\sim$ 150 ms) which changes the state of the DPDT relay.

PC - DP711 power supply: The connection between the PC and the DP711 is made by a RS232 serial interface. The control software in Python, using the pySerial

library [29] again, lets the PC communicate with the DP711 by sending SCPI. Some examples of these commands (available at the Programming Guide [25]) are:

- `*IDN?` to identify the instrument (or check connection)
- `:VOLT:LEV x` to set x volts.
- `:MEAS:CURR?` to read intensity

In this case, the flow of information can be considered unidirectional since the unique commands sent are `:VOLT:LEV x`.

DP711 - Peltier Plate: The two terminals of the DP711 are connected to the terminals of the Peltier plate, but passing through the DPDT relay, in the switching-polarity mode. By this setup, the tension applied to the Peltier can be positive or negative, causing to heat or cool the sample.

Table 3.2: Summary of communication channels in the control pipeline.

Channel	Protocol/Interface	Purpose
MAX31865 RTD → Arduino	SPI	High-precision temperature measurement
Arduino ↔ PC	USB Serial	Temperature data exchange and relay control commands
PC → DP711 power supply	RS232, SCPI	Voltage setpoint control for the Peltier module
Arduino → DPDT Relay	Digital Output	Polarity switching for the Peltier module

3.3 Software implementation in Python

The control software was completely developed in Python, managing all the connections, readings, and calculations in one script. Everything is contained in the class `TemperatureController`, defined in `main.py` and called in `run_controller.py`. The codes are available on GitHub[2] for further inspection.

The standard Python libraries used in the code were:

- `time`: for managing delays and time measurements.
- `serial` (`pyserial`): for serial communication with the Arduino and the power supply.

[2]The full source code for the temperature control system, including the Arduino sketch, Python scripts and documentation, is available at: https://github.com/menendezdiaz/TemperatureController.git

- `matplotlib.pyplot`: for real-time data visualization.
- `matplotlib.widgets`: for interactive GUI elements (e.g., buttons and text boxes).
- `numpy`: for numerical computations and data processing.
- `collections.deque`: to store a bounded history of errors for the integral term in the PID controller, mitigating windup.
- `atexit`: to ensure proper shutdown routines (e.g., resetting the output voltage or closing serial ports).

3.3.1 Control logic

The control logic is principally implemented in the `run()` method, inside the class `TemperatureController` in `main.py`. The chronological sequence for each iteration is:

1. Reads temperature from Arduino via serial communication
2. Calculates the PID output from the user's setpoint
3. Sets a voltage to the power supply with SCPI commands via serial communication
4. Decides whether to heat or cool with the relay via Arduino, based on thresholds
5. Creates a graphic interface

The are some additional control features, which include:

- Configuration for PID parameters, computed in `FIT_PID_PARAMS.py`
- Power factor and dynamic damping for heating, to reduce efficiency and avoid overshooting
- Anti-windup based on limiting the integral window (`deque`)
- Offset correction by rearranging setpoint temperature, contained in the code `CORRECTED_SETPOINT.py`

3.3.2 Interface and user interaction

The graphic user interface (GUI) (Figure 3.4) allows to test the controller and change different parameters in real time. It was integrated using `matplotlib.pyplot` `matplotlib.widgets` libraries.

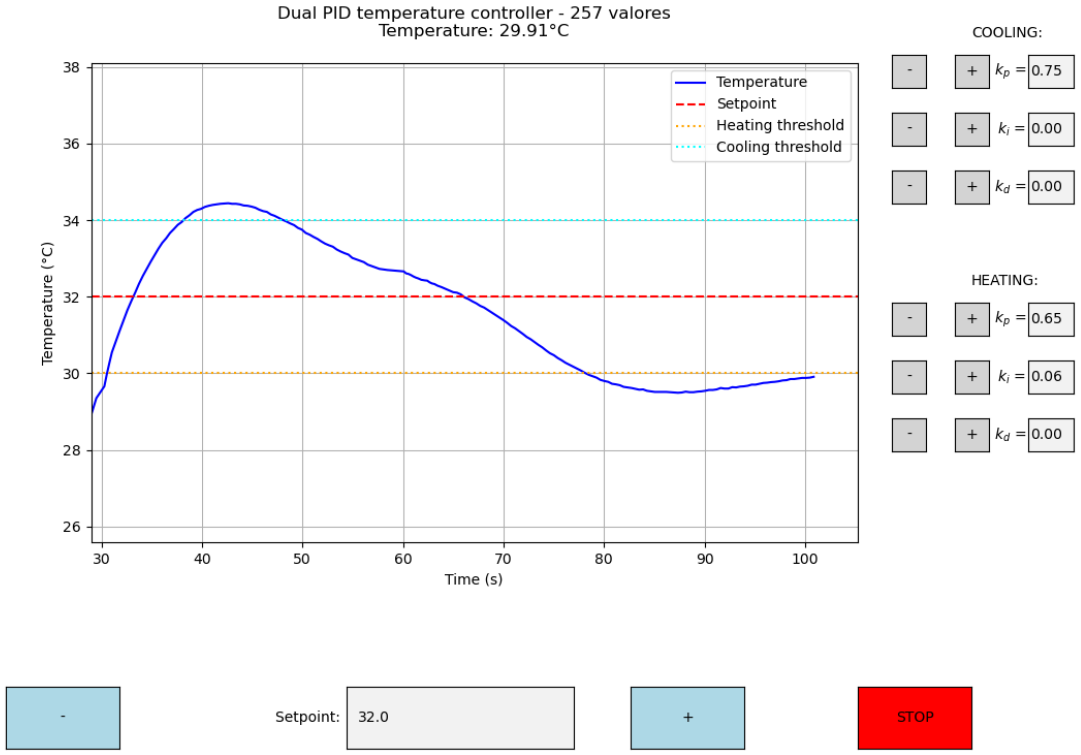

Figure 3.4: User interface (GUI) with visual representation, and modifiable setpoint and PID parameters

The graph shows: temperature in real time, setpoint, and hysteresis thresholds (optional). The buttons allow to: change temperature setpoint, change dual PID parameters, and stop the loop safely. The code also permits the user to generate other plots such as voltage vs time, voltage vs temperature, and other processing features.

Chapter 4

Experimental results and PID tuning

En este capítulo se presentan y se analizan los resultados obtenidos, incluyendo estabilidad térmica, tiempo de respuesta y comparaciones. Para obtener dichos resultados, se requiere una puesta a punto del controlador. Esto se traduce por un ajuste de los parámetros del controlador PID, que en este caso en concreto se ha realizado mediante un ajuste de datos experimentales que varían los parametros según la temperatura deseada, y algunas correcciones extra para terminar de afinar la respuesta.

The sections below present the experimental results of the temperature measured by the sensor, $T_{measured}$, vs time, in several tests of the controller for various temperature setpoints τ. To get those results, it was necessary to have a specific strategy for adequate tuning of the controller. This means setting the PID parameters, which completely changes the response of the system, and adding some extra corrections to get the exact desired behavior. Therefore, a section for this strategy precedes the final experimental results.

4.1 Controller tuning and calibration

PID controllers in general require a deliberated selection of the parameters, known as tuning. A properly tuned PID controller ideally takes the system response to the setpoint in the shortest time possible, minimizing oscillations, but in practice, the selection could be non-trivial and could even require additional parameters to accomplish system demands.

4.1.1 Dual PID and gain scheduling

The controller requires an appropriate tuning of the PID control parameters, following the standard rules detailed in section 2.3.1. Several tests were necessary to refine the values of K_p, K_i, K_d, in which it was finally found that these parameters could not be fixed for every temperature setpoint τ. This peculiarity might be the result of the non-linear and asymmetric behavior of the system, so the strategy implemented

was to create a **dual PID**, where the algorithm differentiates whether the system is in heating or cooling mode. In this way, not only were different combinations of K_p, K_i, K_d defined, but also a power factor, in order to reduce the voltage when heating to balance with the cooling efficiency.

Then, it was also found that for τ near the room temperature, the system could not stabilize $T_{measured}$, unless a P-controller was considered (instead of a PID controller), where

$$K_i = K_d = 0 \iff 18 \lesssim \tau \lesssim 25°C$$

At other temperatures, a P-controller might work, but the steady-state error would be excessive. The **best values of K_i and K_d observed were larger for lower** (higher) **temperatures in cooling** (heating) **mode**, and approximately **constant for K_p**. Actually, there is a big set of combinations of K_p, K_i, K_d which worked well for a fixed temperature τ. This brings forth the idea of an adaptive and automatically variable set of PID parameters, a strategy known as **gain scheduling** [30]. By this method, the algorithm will set itself a combination of K_p, K_i, K_d which work well enough, for a selected τ, resulting in six functions: $K_p(\tau)$, $K_i(\tau)$ and $K_d(\tau)$ for cooling mode and another three for heating mode.

The process to get $K_p(\tau)$, $K_i(\tau)$ and $K_d(\tau)$ is detailed in Appendix A. These effective functions successfully achieve the main goal of the system - to have thermal stability, although the steady-state error is not minimized. To avoid instability, sub-damping or over-oscillations, K_p, K_i and K_d were underestimated in each measurement. The final operating range values for these parameters are shown in Table 4.1.

Table 4.1: Range of values for the 6 effective control functions used in the PID system.

K_p^{cool}	$K_i^{\text{cool}}(\tau)$	$K_d^{\text{cool}}(\tau)$	K_p^{heat}	$K_i^{\text{heat}}(\tau)$	$K_d^{\text{heat}}(\tau)$
0.75	0 – 0.50	0 - 0.03	0.65	0 - 0.3	0 - 0.02

4.1.2 Offset correction

In order to get rid of the offset or steady-state error, another function was implemented, which allows the user to collect data from the observations and install a correction in the setpoint sent to the controller. In the laboratory conditions, the average offset values for different setpoints are (Table 4.2):

Table 4.2: Measured temperature offset as a function of the setpoint for particular conditions in the laboratory.

Setpoint ($°C$)	5	10	15	20	25	30	35	40
Offset ($°C$)	+2.1	+1.8	+1.6	+0.6	-0.5	-1.1	-1.7	-2.0

With these values, a linear fit was done (detailed in Appendix B) so the function sets a correction in the setpoint introduced to the control algorithm. This *sketchy* but effective solution allows the user to get almost the exact desired temperature,

but it will depend on the conditions of the room, the efficiency of all components, the type of insulation, etc.

4.2 Results

According to the objectives - get **stable temperature** and **minimize the steady-state error** with the setpoint τ -, the results of the performance are summarized in different cases. To test the controller, due to the requirements of the experiment, a few minutes are enough, so the results are shown for 5 minutes on average.

In order to analyze the stability of the future results of this system in particular, we should first measure the room temperature behavior, that will establish the mean background fluctuations. In this context, we can test stability considering the maximum fluctuations, introducing a parameter ΔT_{300} defined as the distance between maximum and minimum temperatures in a time window of 5 minutes (300 seconds); this is:

$$\Delta T_{300s} \equiv T_{\max} - T_{\min}$$

4.2.1 Room temperature behavior

The laboratory temperature fluctuations will affect the temperature set by the controller, as the sample requires not to have any coverage. The stability of the signal will be determined by the room disturbances of air, but also by the sampling period and resolution of the sensor.

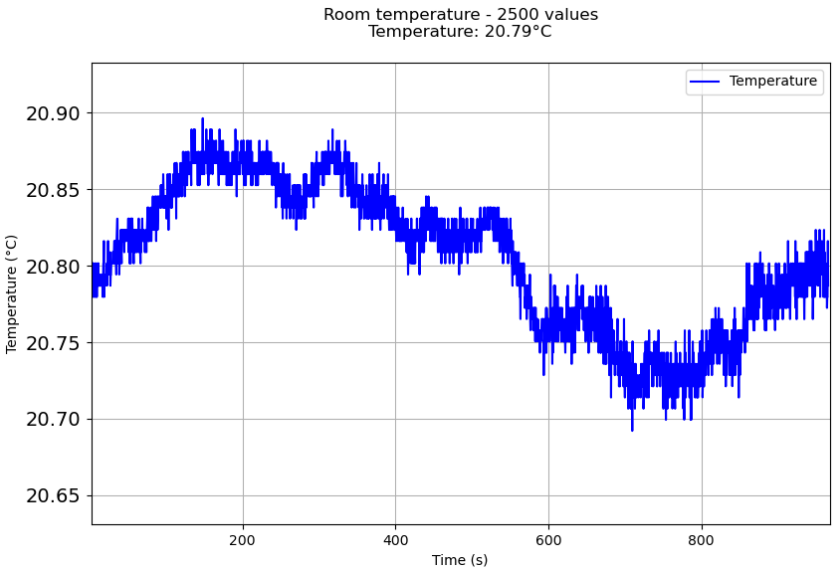

Figure 4.1: Room temperature fluctuations in 15 minutes. Without any coverage the sample will experience a maximum fluctuation of $\Delta T_{300s} \approx 0.1°C$

.

The measurements are shown in Figure 4.1. There are two contributions to the behavior of the curve, the sensor noise, which causes the small and fast oscillations,

and the external disturbances, which cause big and slow oscillations. The sensor noise can be approximated to the width of the curve, less than $0.05°C$; the external disturbances are around $0.1°C$, but any abrupt change in the room dynamic can cause much larger ΔT.

4.2.2 Response with gain-scheduled PID

The results of the temperature given by the controller for different fixed setpoints are separated in 3 cases: cold, mid and hot setpoints.

For the cold setpoint $6°C$ were selected, shown in Figure 4.2. The final temperature was stable by $\Delta T_{300s} \approx 0.2°C$ and did not reach the setpoint for $1.3°C$.

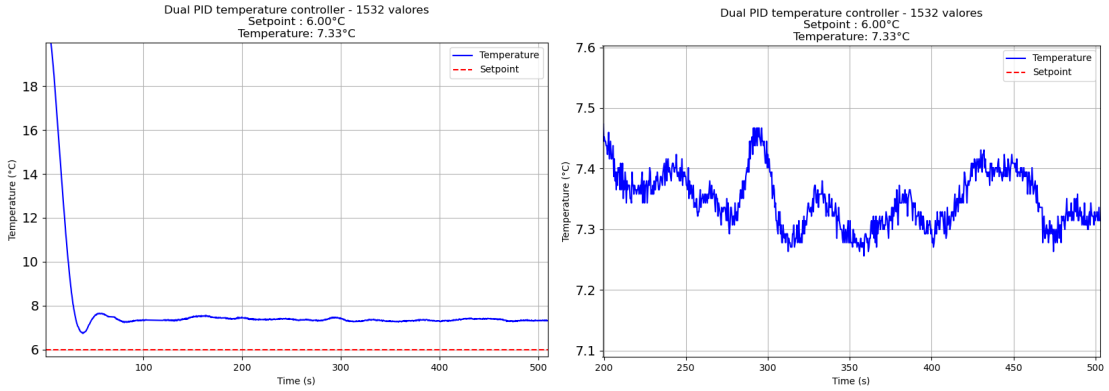

Figure 4.2: Temperature vs time for $\tau = 6°C$. On the left, the general view, with an offset around $1.3°C$. On the right, the zoomed image ($\Delta T_{300s} \approx 0.2°C$).

For the mid setpoint $18°C$, Figure 4.3 shows that the temperature is more stable, and the offset is smaller, around $0.7°C$.

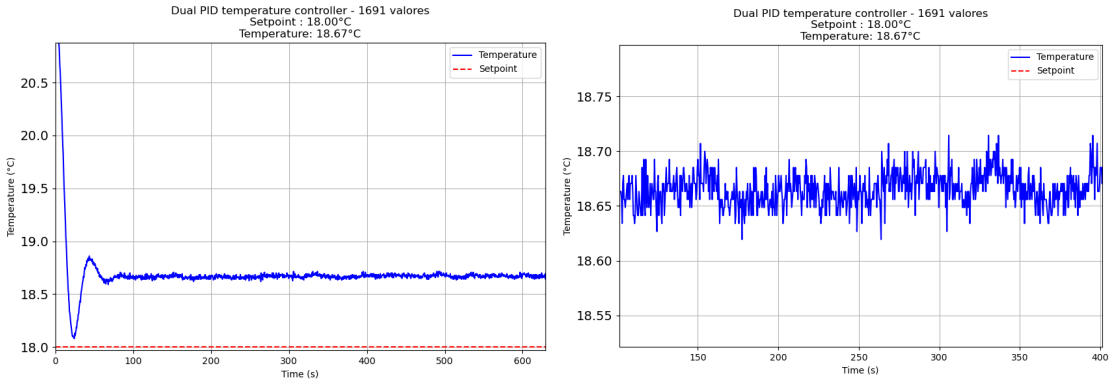

Figure 4.3: Temperature vs time for $\tau = 18°C$. On the left, the general view, with an offset around $0.6°C$. On the right, the zoomed image ($\Delta T_{300s} \approx 0.1°C$).

Finally, Figure 4.4 shows the hot setpoint test. Here, the plate needs to heat, so the offset is negative, around $-2.2°C$. Oscillations are larger again, almost about $0.3°C$.

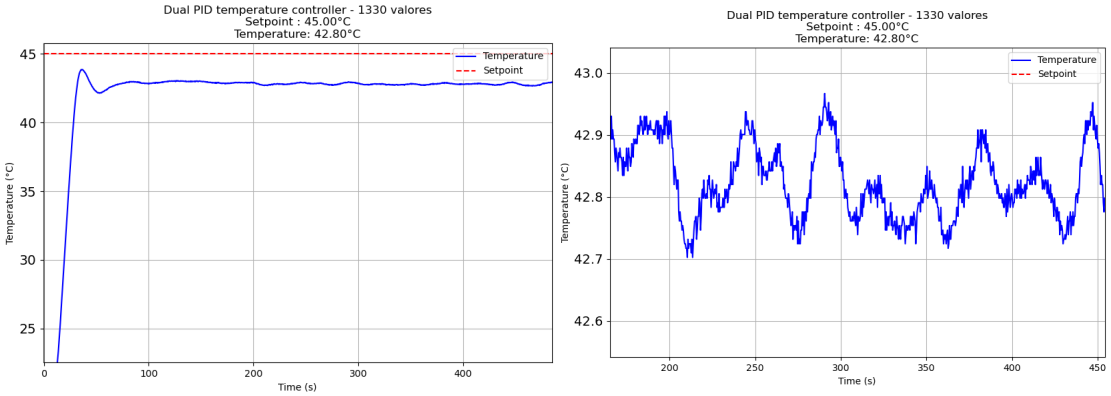

Figure 4.4: Temperature vs time for $\tau = 45°C$. On the left, the general view, with an offset around $1.2°C$. On the right, the zoomed image ($\Delta T_{300s} \approx 0.2°C$).

For temperatures near the room temperature the oscillations are almost equal to 4.1, but far from there, the oscillations become bigger due to instability for being far from the equilibrium.

4.2.3 Response with offset-correction function

If the user wants to reduce the steady-state error, an additional empirical function was implemented to be used. This function, by introducing the offset observed in different setpoints, will rearrange the τ introduced to the PID algorithm. This correction will work for a specific set of conditions, and should be calibrated before the desired tests. In the laboratory conditions, introducing the data from Table 4.2, the resulting measurements are shown in Figure 4.5. The goal here is to test the new offset, so the time window required to get stability will be enough.

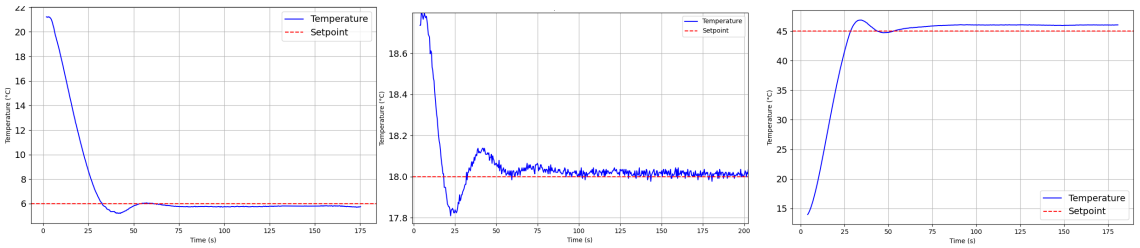

Figure 4.5: Temperature response for different setpoints, using the setpoint correction which almost eliminates the steady-state error.

4.2.4 Results with thermal insulation

Due to the fluctuations caused by the external disturbances, the idea was to try the same measurements with some kind of insulation which could enhance thermal integration between the sample, the sensor, and the test face of the Peltier. Although the experiment still requires not to have any coverage, if any insulation method is implemented, this section will show how the controller would work. The cover implemented was far from being ideal but it will let us have an idea how recommendable

it would be to design a compatible insulation. Similar test setpoints were tried to compare with the results, but no offset correction was applied to see how the raw controller works with insulation.

First was tested with the same cold (Figure 4.6), mid (Figure 4.7) and hot (Figure 4.8) setpoints.

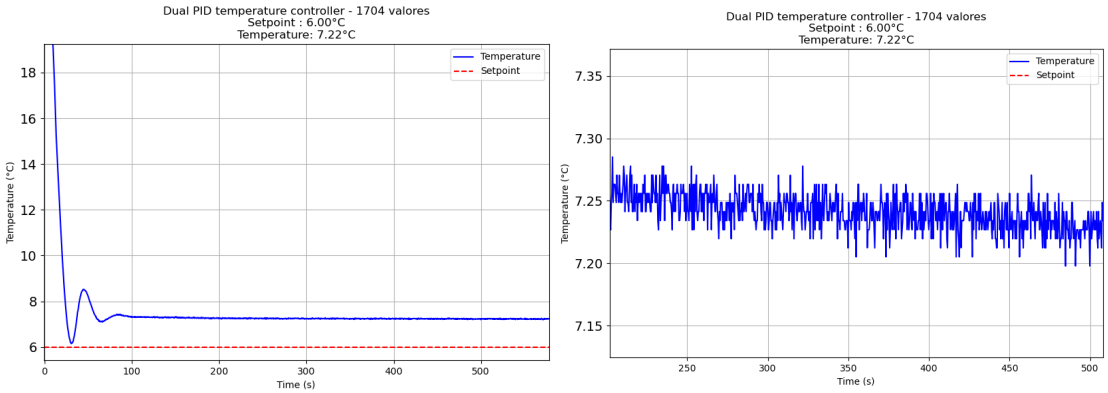

Figure 4.6: Temperature vs time for $\tau = 6°C$. On the left, the general view, with an offset around $1.2°C$. On the right, the zoomed image ($\Delta T_{300s} < 0.1°C$).

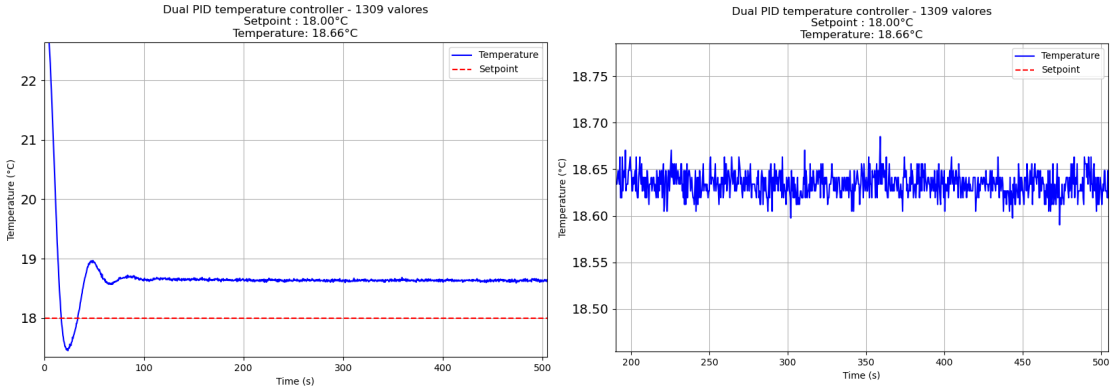

Figure 4.7: Temperature vs time for $\tau = 18°C$. On the left, the general view, with an offset around $0.6°C$. On the right, the zoomed image ($\Delta T_{300s} < 0.1°C$).

With respect to the downward trend ($0.05°C/300s$) in Figure 4.6 (right), it can be caused by slow room temperature oscillations, which are the result of any small disturbance in the room, such as opening a door, a cold breeze, someone leaving, etc. The insulation reduces these effects, but it is not perfect yet.

In all cases, the offset was significantly reduced, but the most relevant difference is the stability. In the three setpoints, we have $\Delta T_{300} \leq 0.1°C$, which is the result of thermal insulation. That fluctuations can almost be considered sensor noise.

4.2.5 General functioning and relay action

In general, the user is able to select the temperature setpoint τ in real time. Figure 4.9 shows the behavior of the temperature by changing τ. The results are the same

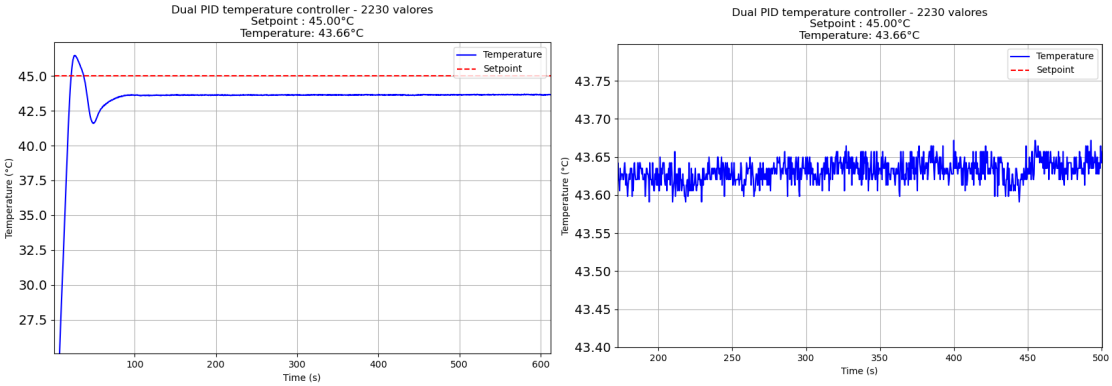

Figure 4.8: Temperature vs time for $\tau = 45°C$. On the left, the general view, with an offset around $1.3°C$. On the right, the zoomed image($\Delta T_{300s} < 0.1°C$).

as before but for big changes the oscillations last a little longer, and here is where the relay is useful. When switching from a cold to a hot temperature or vice versa, the relay must switch the Peltier polarity. The cooling and heating thresholds are set to retard the action in both modes due to thermal hysteresis. Without these thresholds the relay could be constantly switching or could cause over-oscillations.

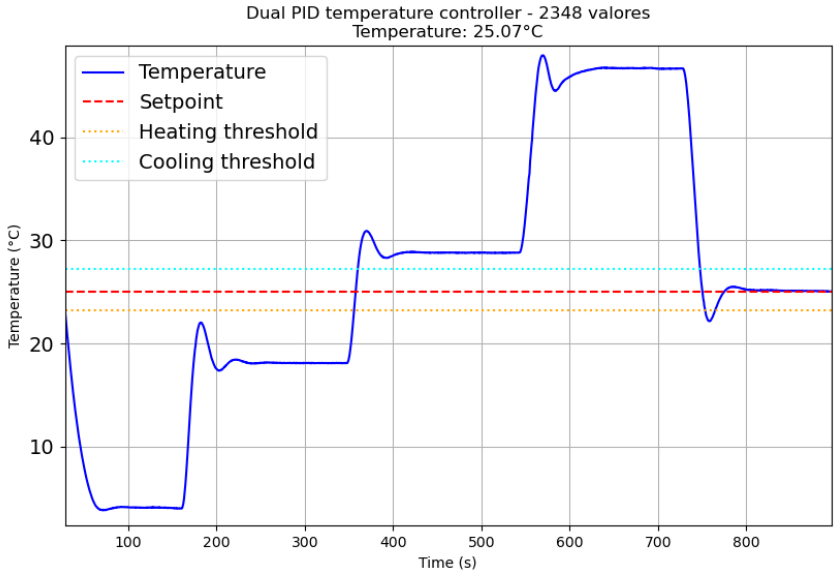

Figure 4.9: Temperature response for various setpoint changes while running. The setpoint and thresholds lines correspond to the last setpoint introduced by the user.

The overshoot is sometimes notable, caused by the switching of cold/hot modes in big changes of τ. In the last setpoint of Figure 4.9, is clear how the temperature rate changes its sign when it crosses the cooling threshold, due to the presence of the relay. Then it continues in heating mode because the setpoint is above room temperature.

4.2.6 Summary of results and comparison

To check the results there is Table 4.3, which summarizes the sections above, and allows to compare offset and fluctuations in different cases for different setpoints. The best results were obtained with insulation, were stability is more than acceptable. Offset is probably caused by the lack of precision, insulation and perfect thermal integration between the plate, sensor and sample; but it should not be a problem, since the main objective is to have stable temperature.

Table 4.3: Effect of setpoint correction and thermal insulation on temperature offset and fluctuation.

Setpoint ($^{\circ}C$)	**Offset (no correction)**	**Offset (correction)**	ΔT_{300s} **(no insulation)**	ΔT_{300s} **(insulation)**
6	$+1.3$	~ 0.2	~ 0.2	< 0.1
18	$+0.7$	~ 0.1	~ 0.2	< 0.1
45	-2.2	~ 0.3	~ 0.2	< 0.1

Chapter 5

Discussion and conclusions

5.1 Summary of achievements

This project successfully developed a temperature control system that meets the stringent requirements of a physics experiment, combining a **dual PID controller with gain scheduling**, an **offset correction mechanism**, and a **user-friendly GUI interface**. The dual PID approach enabled distinct tuning for heating and cooling modes, addressing the inherently asymmetric behavior of the Peltier thermoelectric module. By scheduling different PID gains depending on the temperature range (cooling below ambient and heating above ambient), the controller maintained stable operation in both modes without excessive oscillation or overshoot.

As a result, the system achieved high temperature stability: in an open-air configuration (no added insulation) the temperature could be held constant within a fluctuation band of approximately $0.2°C$ peak-to-peak, and this was further improved to about $0.06°C$ when a simple thermal insulation was applied. These figures correspond to maintaining the sample's temperature essentially within $\pm 0.1°C$ in free air and $\pm 0.03°C$ with insulation - a remarkable stability given the experimental constraints.

Moreover, the implementation of an **empirical offset correction function** effectively eliminated steady-state errors. With this correction, the temperature output closely matched the desired setpoint across the tested range, even in scenarios (such as extreme low or high setpoints) where an uncompensated system would otherwise exhibit a bias of up to $1°C$.

Finally, a graphical user interface was developed to allow interactive control and monitoring of the system. Through the GUI, users can input setpoints in real time, visualize temperature trends, and adjust parameters, making the controller **highly accessible and practical for laboratory use**.

5.2 Relevance of the obtained results

The results obtained are highly significant in the context of the experiment's requirements and constraints. **Importantly, the controller satisfies the key constraint of operating without insulation in a normal lab environment**,

which was necessary because the experimental setup did not allow enclosing the sensor-Peltier-sample area.

Even **exposed to ambient air**, the system maintained temperature stability on the order of a few tenths of a degree. Over short measurement intervals (around 5 minutes), the sample's temperature remained stable within $0.2°C$. This level of stability is largely governed by unavoidable room disturbances such as air currents and ambient fluctuations.

The inclusion of a **gain-scheduled dual PID** was instrumental in this regard: because Peltier devices heat and cool with very different efficiencies, using separate PID parameters for each mode ensured optimal response in both directions. A small **hysteresis band** was deliberately built into the control logic when switching between heating and cooling, preventing high-frequency toggling and maintaining smooth temperature control.

Another critical outcome is the **improved temperature accuracy** due to offset reduction. Initially, an offset of up to $2°C$ was observed under open-air conditions. After correction, the system delivered temperatures with negligible steady-state error. Additionally, when a modest insulating cover was introduced, both the offset and fluctuation range were further reduced.

Although the system's stability ($0.03°C$ with insulation) is slightly lower than that of high-end commercial devices (typically around $0.002°C$ over 24 hours), it is **more than sufficient for short-term experimental applications**. The lack of insulation represents a more demanding scenario, so achieving this level of performance confirms the robustness and reliability of the implemented strategies.

5.3 Future perspectives

Building on the current achievements, several improvements and extensions could enhance the system's performance:

- **Developing compatible thermal insulation:** Even though the setup must remain largely exposed, a partial or transparent enclosure could reduce convective losses and further increase stability.
- **Extending offset calibration:** The correction function could be expanded to include a wider range of ambient conditions and setpoints, adapting to different thermal loads or lab environments.
- **Implementing advanced auto-tuning:** Algorithms such as Ziegler–Nichols or adaptive model-based methods could automate PID tuning and adjust dynamically to changes in the system or surroundings.
- **Relay-based system identification:** Including bi-directional relay feedback testing would enable automated characterization of the system's dynamics and self-calibration of PID parameters.
- **Enhancing the GUI:** Adding features like data logging, remote access, or visualizations of control signals could improve usability and integration with experimental workflows.

- **Long-term stability testing:** Extended runs (e.g., 24 hours) would allow analysis of drift and reliability, revealing the need for possible recalibrations or hardware improvements.

In conclusion, the developed controller constitutes a solid foundation for precise temperature regulation in laboratory environments. The combination of practical constraints, software design, and control theory provides a valuable and adaptable tool for research. With further refinement, this platform can evolve into a general-purpose temperature control solution for a wide range of experimental applications.

Appendix A

PID tuning and gain scheduling

The goal was to set an auto-adjustable set of parameters depending on the temperature set by the user τ. In particular, the final decision was to tune the PID for a number of temperatures in the operating range (e.g., 3, 6, 9, 12, 15, 18 °C) on different days, and take the values which achieved best observed control performance. Then these values were averaged and used for a data fit. For each value of temperature, a thorough and deliberated tuning was carried out, being as rigorous as possible, but considering the environmental fluctuations. These limitations bring an error associated with each point in the experimental measurements of $K_p(\tau)$, $K_i(\tau)$ and $K_d(\tau)$.

The resultant functions used are summarized in Figure A.1.

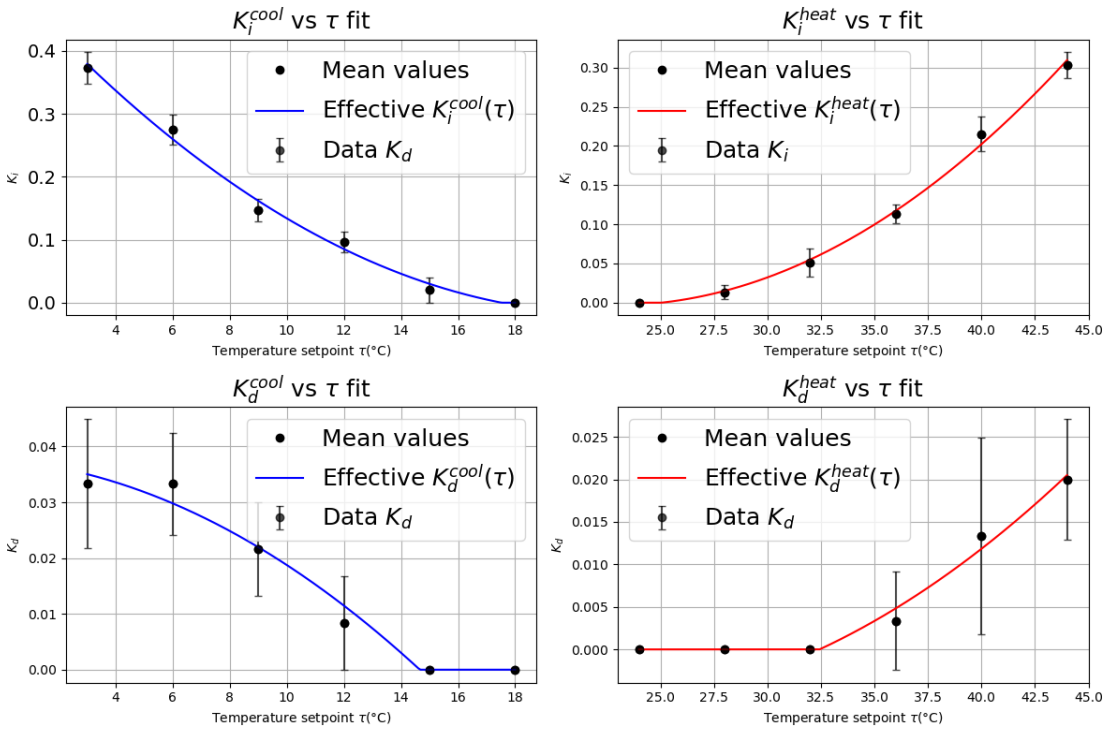

Figure A.1: The effective K_i and K_d for both modes depending on temperature

The K_p was maintained constant, since the proportional gain did not seem to

be strongly dependent on temperature. In contrast, $K_i(\tau)$ and $K_d(\tau)$ were obtained from data in Table A.1 using NumPy's polynomial fitting functions (`numpy.polyfit` and `numpy.polyval`) for a second degree curve. These fitted functions are symbolic, as they are intended to be empirical approximations derived from manually selection and visually guided tuning.

Table A.1: Mean values of K_p, K_i and K_d for each setpoint temperature.

Temperature	K_p	K_i	K_d
3 °C	0.75	0.37	0.03
6 °C	0.75	0.28	0.03
9 °C	0.75	0.15	0.02
12 °C	0.75	0.10	0.01
15 °C	0.75	0.02	0.00
18 °C	0.75	0.00	0.00
24 °C	0.65	0.00	0.00
28 °C	0.65	0.01	0.00
32 °C	0.65	0.05	0.00
36 °C	0.65	0.11	0.00
40 °C	0.65	0.21	0.01
44 °C	0.65	0.30	0.02

Appendix B

Offset correction

The offset correction, implemented by rearranging the setpoint employed by the control algorithm, was made in a qualitative, non-systematic and heuristic way. This is due to the unstable and unpredictable behavior of the system. The used results are summarized in Figure B.1, using the data from Table 4.2.

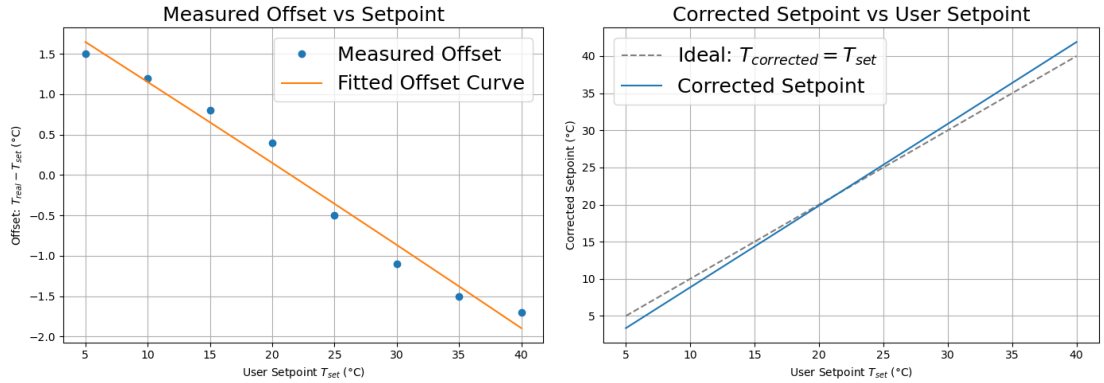

Figure B.1: On the left, the fitted experimental points for offset vs temperature; points have no error because this is a rough approximation. On the right, T_{measured} vs τ - the blue line represents the rearranged setpoint.

List of acronyms

RTD resistance temperature detector

PT100 platinum-based RTD with 100 ohms resistance at $0°C$

PID proportional-integral-derivative controller

K_p proportional gain

K_i integral gain

K_d derivative gain

DPDT double-pole-double-throw relay module

DP711 model name of the power supply used

SCPI standard commands for programmable instruments

PC personal computer

GUI graphic user interface

τ **(tau)** Temperature setpoint (target)

ΔT_{300s} Temperature fluctuation parameter. The difference between maximum and minimum temperatures measured over a 300-second (5-minute) time window.

Bibliography

[1] L. L. Martín, C. Pérez-Rodríguez, P. Haro-González, and I. R. Martín, "Whispering gallery modes in a glass microsphere as a function of temperature," *Opt. Express*, vol. 19, pp. 25792–25798, Dec 2011.

[2] K. Ogata, *Modern Control Engineering*. Upper Saddle River, NJ: Prentice Hall, 5th ed., 2010.

[3] Omega Engineering, *RTD Temperature Measurement*. 2020. Available online: `https://www.omega.com/temperature/Z/pdf/z021-032.pdf`.

[4] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and A. S. Lavine, *Fundamentals of Heat and Mass Transfer*. Hoboken, NJ: Wiley, 8th ed., 2017.

[5] H. K. Khalil, *Nonlinear Systems*. Upper Saddle River, NJ: Prentice Hall, 3rd ed., 2015.

[6] A. Bejan, "Convection heat transfer," 2013.

[7] V. A. Drebushchak, "The peltier effect," *Journal of Thermal Analysis and Calorimetry*, vol. 91, no. 1, pp. 311–315, 2008.

[8] M. K. Shilpa, M. A. Raheman, A. Aabid, M. Baig, R. K. Veeresha, and N. Kudva, "A systematic review of thermoelectric peltier devices: Applications and limitations," *Fluid Dynamics & Materials Processing*, vol. 18, no. 3, pp. 1–12, 2022.

[9] W. D. Callister and D. G. Rethwisch, *Materials Science and Engineering: An Introduction*. Hoboken, NJ: Wiley, 10th ed., 2018.

[10] M. S. Van Dusen, "Platinum resistance thermometry at low temperatures," *Journal of the American Chemical Society*, vol. 47, no. 2, pp. 326–332, 1925.

[11] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[12] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and R. D. Raut, "A review of pid control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, no. 2, pp. 818–827, 2021.

[13] A. O'Dwyer, *Handbook of PI and PID Controller Tuning Rules*. London: Imperial College Press, 3rd ed., 2009.

[14] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle, *Process Dynamics and Control*. Hoboken, NJ: Wiley, 4th ed., 2016.

[15] S. Skogestad, "Simple analytic rules for model reduction and pid controller tuning," *Journal of Process Control*, vol. 13, no. 4, pp. 291–309, 2003.

[16] O. Garpinger, T. Hägglund, and K. J. Åström, "Performance and robustness trade-offs in pid control," *Journal of Process Control*, vol. 24, no. 5, pp. 568–577, 2014.

[17] K. J. Åström and T. Hägglund, "Advanced pid control," *ISA - The Instrumentation, Systems and Automation Society*, 2006.

[18] K. J. Åström and L. Rundqwist, "Integrator windup and how to avoid it," in *Proceedings of the 1989 American Control Conference*, pp. 1693–1698, IEEE, 1989.

[19] A. Visioli, "Modified anti-windup scheme for pid controllers," *IEE Proceedings - Control Theory and Applications*, vol. 150, no. 1, pp. 49–54, 2003.

[20] C. A. Smith and A. B. Corripio, "Principles and practice of automatic process control," *Wiley*, 2002.

[21] Hebei I.T. (Shanghai) Co., Ltd., "Thermoelectric cooler tec1-12706." `https://peltiermodules.com/peltier.datasheet/TEC1-12706.pdf`, n.d. Rev. 2.03.

[22] R. M. Suares and et al., "Cooling machine adapted peltier cooling module tec1-12706 to understand heat transfer application," *BIO-MEKANIK Journal*, 2024. Describes the use of TEC1-12706 for experimental heat transfer applications with PSU and fan.

[23] A. Kherkhar and et al., "Thermal investigation of thermoelectric device based on arduino and pid control approach," *SSRN Electronic Journal*, 2022. Demonstrates PID-based temperature control using TEC1-12706 with Arduino.

[24] Rigol Technologies, *DP711 Programmable DC Power Supply*, 2020. Available online: `https://www.rigolna.com/products/dc-power-loads/dp700/`.

[25] Rigol Technologies, *DP700 Series Programmable Linear DC Power Supply Programming Guide*, 2016. Available online.

[26] Texas Instruments, "Driving a peltier element (tec): Efficiency and aging," tech. rep., Texas Instruments, 2021. Available online.

[27] Arduino, "Arduino uno rev3," 2020. Available online: `https://store.arduino.cc/products/arduino-uno-rev3`.

[28] Adafruit Industries, "Adafruit max31865 rtd pt100 or pt1000 amplifier," 2020. Available online: `https://www.adafruit.com/product/3328`.

[29] pySerial Developers, *PySerial: Welcome to pySerial's documentation*, 2020. Retrieved November 30, 2021, `https://pyserial.readthedocs.io/en/latest/`.

[30] K. J. Åström and B. Wittenmark, *Adaptive Control*. Dover Publications, 2nd ed., 2008.