universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Developing a Temperature Control System for Researching Global Brain Dynamics in C. elegans

verfasst von | submitted by

Verena Steinkasserer BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 878
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Verhaltens-, Neuro- und Kognitionsbiologie
Betreut von | Supervisor:	Univ.-Prof. Dr. Manuel Zimmer

Acknowledgements

I am truly grateful for the opportunity to do my master's thesis in the Zimmer lab. Being a part of the Zimmer Group has been an incredibly enriching experience. I am very grateful for the wonderful people I've met, the professional growth and the invaluable friendships.
I am particularly grateful to my supervisor Manuel for giving me the incredible opportunity to work on this project. His support and enthusiasm not only inspired me but also motivated and encouraged me.
Special thanks to Lukas for his contribution and advise, which have been fundamental to my learning and have been driving this project forward. Thanks to Pedro for teaching me lab techniques and for his effort and dedication. I would also like to express my deepest appreciation to Richard, Itamar, Fabio, Josefine, Samuel, Jalaja, Johannes, Laxmi, Charlie and all the other current and past lab members of the lab for their scientific guidance and personal support, for the fun and memorable times.
A deep thank you to my family and friends for their endless support and love. To Dibi a very special thank you. I couldn't have done this without you, you mean everything to me.
To everyone who has been a part of this journey - thank you for making it unforgettable.

Abstract

In neuroscientific research on the model organism *Caenorhabditis elegans* (*C.elegans*), it is important to control environmental conditions to ensure accurate and reliable experimental results. This is especially true in experiments where environmental factors, such as temperature, significantly influence or alter behaviors and/or neuronal activities under study.
The temperature-sensitive mutation of the *cha-1* gene, which codes for choline acetyltransferase (ChAT), affects acetylcholine (ACh) synthesis at different temperatures. Specifically, it leads to disruption at higher temperatures and normal functioning at lower temperatures. To investigate the effects of ACh's perturbation on global neuronal dynamics, a temperature control system (TCS) was developed. It is designed to manage precise environmental temperature conditions during experiments.
The system operates on the principles of closed-loop feedback systems. Key components of the TCS include temperature sensors, actuators (Peltier elements), and a microcontroller unit for real-time temperature adjustments. The system uses a PI (Proportional-Integral) control algorithm to dynamically adjust temperatures with minimal deviation from set points based on feedback from the system being controlled. The performance of the system was assessed through step response experiments, where the TCS ability to adjust and maintain the desired temperatures is tested.
The proportional and integral components are tuned for optimal system responsiveness and steady-state accuracy. The system demonstrates adaptive responses in both cooling and heating conditions across different set point changes. The system's thermal properties (particularly the thermal inertia of Polydimethylsiloxane (PDMS), a silicon-based polymer), influence the response curve, with a rather slow adaptation to control inputs due to the low thermal conductivity of materials used in the microfluidic device. Despite its slower response, the system maintains stable set temperatures within a defined tolerance range after reaching the settling point.

Zusammenfassung

In neurowissenschaftlichen Forschungen am Modellorganismus *Caenorhabditis elegans (C. elegans)* sind kontrollierte Umweltbedingungen entscheidend, um Validität, Zuverlässigkeit und Glaubwürdigkeit der experimentellen Ergebnisse zu gewährleisten. Dies gilt insbesondere für Experimente, bei denen Umweltfaktoren wie Temperatur das Verhalten und/oder neuronale Aktivitäten der untersuchten Organismen signifikant beeinflussen oder verändern können.
Die temperatur-sensitive Mutation des *cha-1*-Gens, das für Cholinacetyltransferase (ChAT) kodiert, beeinflusst die Synthese von Acetylcholin (ACh) bei unterschiedlichen Temperaturen. Bei höheren Temperaturen kommt es zu Störungen in der Acetylcholin Produktion, während bei niedrigen Temperaturen eine normale Funktion beobachtet werden kann. Um die Auswirkungen dieser ACH Perturbation auf globale neuronale Dynamiken zu untersuchen, wurde ein Temperaturkontrollsystem (TKS) entwickelt. Dieses System ist darauf ausgelegt, während Experimenten die Umgebungstemperatur des Wurms genau zu kontrollieren.
Das System basiert auf den Prinzipien von geschlossenen Regelkreisen. Zu den Schlüsselkomponenten des TKS gehören Temperatursensoren, Aktuatoren (Peltier-Elemente) und eine Mikrocontroller-Einheit zur Echtzeit-Temperaturüberwachung- und regelung. Mithilfe eines Proportional-Integral (PI) Reglers passt das System dynamisch Temperaturen mit minimalen Abweichungen von den Sollwerten an. Diese Anpassungen basieren auf Rückmeldungen aus dem kontrollierten System. Die Leistungsfähigkeit wurde durch Sprungantwortexperimenten bewertet, in denen getestet wurde ob das TKS die gewünschte Temperaturen sowohl einstellen als auch halten kann.
Die proportionalen und integralen Komponenten sind so aufeinander abgestimmt um optimale Systemreaktion und Genauigkeit im stationären Zustand zu erreichen. Das System zeigt adaptive Reaktionen in Kühl- wie auch in Heizbedingungen bei verschiedenen Sollwertänderungen. Die thermischen Eigenschaften des Systems, (insbesondere die thermische Trägheit von Polydimethylsiloxan (PDMS), ein silikonbasiertes Polymer) beeinflussen die Reaktionskurve. Dies äußert sich durch eine eher langsame Anpassung an Steuersignale aufgrund der geringen Wärmeleitfähigkeit der verwendeten Materialien. Trotz seiner langsamen Reaktion hält das System stabile Solltemperaturen innerhalb eines definierten Toleranzbereichs nach Erreichen der Absetzzeit.

Contents

1 Introduction

Understanding how neurons interact to shape behavior and process information is a key question in neuroscientific research. Neurons, the basic building blocks of the nervous system, form networks and circuitries that process sensory information, drive motor outputs, and give rise to cognitive functions such as learning and memory. Research into how neurons interact to shape behavior and process information involves studying these neural network dynamics and analyzing their activity patterns. The nematode *Caenorhabditis elegans (C. elegans)* is a valuable model organism for such studies. All the 302 neurons and their connections, known as the connectome, have been mapped which enables detailed studies of neuronal network functions[1]. Despite its simplicity, *C. elegans* exhibits a range of behaviors, from feeding to navigating its environment. Variety of behaviors coupled with a simple nervous system, makes it an ideal model for studying the neural basis of behavior. Another advantage of *C. elegans* as a model organism, is its genetically tractable, this means it is possible to perform precise genetic manipulations to study the effects of specific mutations. Further, its transparency makes it optical accessible which enables direct observation of neural activity within the living worm. This means, it is possible to visualize and capture the dynamic processes of the nervous system in real-time, from neuron-to-neuron communication to network-wide neuronal activity[2].

Brain-wide calcium imaging experiments have previously shown that a large number of neurons in the *C. elegans* nervous system engage in so called global brain dynamics. It is suggested that these dynamics coordinate complex behaviors of the worm. They are characterized by coordinated and synchronized network activities that evolve cyclically (repetitive patterns over time). Unlike isolated, local neural events, these dynamics represent the collective behavior of neuronal populations across the brain. Such network dynamics are essential for the high-level organization of *C. elegans* behavior. The cyclic pattern of brain states align with the animal's motor command sequences, indicating that they coordinate the movement of *C. elegans*. Particularly, these patterns are thought to be the neural basis for the worm's ability to perform sequences of action, such as crawling forward, crawling backwards or turning. This suggests that global brain dynamics play a key role in how the nervous system processes, integrates and translates information into behavior[3].

Acetylcholine is the most abundant neurotransmitter in *C. elegans*' nervous system, with more than a third of the neurons being cholinergic[4]. It is the major excitatory neurotransmitter at neuromuscular junctions and plays an important role in locomotion and coordination, and neuronal communication[5]. Disturbances in Acetylcholine, can have broad-ranging impacts on neuronal network functions. By examining these effects, we can gain insights into the role of Acetylcholine on global brain dynamics.

1.1 *Cha-1* temperature sensitive mutation

The *cha-1* gene in *C. elegans* encodes for Choline Acetyltransferase (ChAT), the enzyme responsible for the synthesis of Acetylcholine. Duerr et. al.[6] analyzed several *cha-1* mutations, finding that most of the available mutant alleles are hypomorphs. Hypomorphic mutations lead to a reduced, but not completely loss of function of the gene product[7],[6]. For *cha-1*, this means that ChAT is still produced and active, but at lower levels and/or with lower activity than in wild-type animals. This reduced activity can lead to impaired synthesis of Acetylcholine, affecting the worm's behavior[6]. Any complete loss-of-function of *cha-1* alleles are lethal during development. Here, the temperature-sensitive allele *md39* is used, which is assumed to be loss-of-function only at restrictive temperatures. This way, it is possible to bypass the lethal phenotype of a null-mutation with the additional benefit to likely observe acute effects, rather than secondary or indirect ones. The temperature-sensitive phenotype of the *cha-1 md39* mutation is characterized by pronounced neuromuscular impairments at higher, restrictive temperatures (25°C), including uncoordinated movements, slow growth and paralysis. Whereas, at lower, permissive temperatures (15°C), mutants exhibit normal, wild-type like behavior[5]. This temperature sensitivity is hypothesized to stem form the non-covalent association of ChAT with synaptic vesicles, important for efficient Acetylcholine synthesis. At higher temperatures, more ChAT protein dissociates from the synaptic vesicles. In *cha-1* mutants, which

have reduced ChAT activity, the synthesis of Acetylcholine is already compromised. When the temperature increases, the reduction in vesicle-associated ChAT further diminishes the efficiency of Acetylcholine synthesis and loading into synaptic vesicles. This leads to more severe neuromuscular dysfunction at higher temperatures. However, these temperature effects are reversible. When the temperature is lowered, ChAT can re-associate with vesicles, restoring Acetylcholine synthesis to levels closer to wild-type[6].

1.2 Temperature Control System (TCS)

The temperature sensitive nature of *cha-1* mutants, requires an experimental setup capable of fine-tuned temperature regulation. The primary goal of this project is the development and validation of a temperature control system (TCS). This system is designed for real-time adjustment of the environmental temperature of the worm within the PDMS layer of a microfluidic chip (described by [8],[3],[9]) during whole-brain calcium imaging. The aim is to maintain accurate and stable conditions for generating reliable neuronal activity data.

Implementing a TCS for temperature control for whole-brain imaging requires careful arrangement of components to ensure compatibility with the experimental setup. This setup consists of an inverted confocal microscope equipped with a water immersion objective lens (40x) and a PDMS microfluidic device. This device includes a worm immobilization channel and a gas-flow channel with respective in- and outlets, all housed within a PDMS layer on a glass cover slip[10]. During imaging, the lens makes direct contact with the cover glass. To achieve effective temperature control, the TCS must be positioned as close to microfluidic device as possible. Minimizing the air gap improves temperature transfer and reduces thermal lag[11],[12]. However, the TCS must at the same time not interfere with the light path of the microscope. The design must fit within the limited space available in and around the microscope without making significant modifications to the existing setup. The TCS needs to be portable for easy movement so it does not interfere the loading of worms into the microfluidic chip. All components involved, their position and thermal properties potentially influence temperature regulation and have to be considered for successful TCS implementation[8],[10].

Precisely controlling environmental parameters and conditions such as temperature within microfluidic devices has been a central topic of previous studies. The control of temperature is important as it has a great impact on a variety of different biological and chemical processes. It ranges form biological studies and disease modeling to cellular and molecular research techniques. Temperature control in microfluidic devices was used to study the dynamic responses of the microtubule cytoskeleton in a temperature-sensitive mutant of fission yeast to temperature fluctuations[13]. In the context of studying diseases like amyotrophic lateral sclerosis (ALS), microfluidic devices that can maintain specific temperature are used in combination with high-resolution imaging. This allows researchers to monitor protein aggregation over time in *C. elegans*, and gives insights into the progression of neuro-degenerative diseases[14].
Various methods have been developed to achieve precise temperature control. They include external and integrated methods and cover the need for measuring, creation and manipulation of microfluidic environments. External heating sources include techniques like Joule heating, the use of microwaves, and lasers. Integrated systems involve embedded heating elements or cooling mechanisms like Peltier elements. For temperature monitoring, devices like thermocouples and thermistors are used. Besides imaging and modeling, these methods also provide control over microfluidic environments for applications such as thermal cycling for PCR and digital microfluidics that involve generating temperature gradients[15],[16],[17].

In this project, the TCS's design should fulfill several requirements:

1. It must maintain a stable temperature of the direct environment of the worm throughout the duration of experiments.
2. For effective whole-brain calcium imaging of *C. elegans*, the TCS must be fully compatible with the inverted confocal microscope and the PDMS microfluidic device used. The design of the TCS must account for the physical and operational requirements of both the microscope and the microfluidic setup. Temperature regulation must not interfere with high-resolution neuronal activity imaging.

3. It must be interactive and user-friendly. This involves implementing a live-updating graph that displays the current temperature data over time. This makes it possible to visualize and observe real-time changes and trends in temperature. In addition, input fields should be included for users to enter commands, such as new temperature set points or power output values.

1.3 Control Systems

Control engineering covers all aspects of governing a dynamic system and is based on system modeling, feedback, system response, and stability. The principles of control engineering are applicable to many different engineering disciplines, such as mechanical, chemical, aeronautical, civil, and electrical. Control systems are designed to regulate the behavior of a system or process (plant) to achieve desired outcomes or responses[18, p.2],[19, p.11-13],[20, p.17],[21, p.60],[22, p.16-17]. For instance, a plant could be a mechanical, electrical or thermal system[23, p.11]. The cruise control and a home thermostat are common examples for control systems [19, p.15-17]. Both systems automatically maintain a set value, vehicle speed and indoor temperature, respectively, through continuous feedback and adjustments. Cruise control keeps a car at a constant velocity despite of disturbances (changes in slope), by measuring the speed of the car and adjusting the throttle accordingly[20, p.65-66]. Meanwhile, a thermostat regulates indoor temperature by activating heating or cooling based on the difference between the actual and desired temperatures. Both systems represent the core principle of control engineering which is achieving a desired outcome by automatically modifying the behavior of a system[18, p.2],[19, p.11-13]. There exist two different approaches within control systems to achieve this aim: Open-loop and closed-loop control.

1.3.1 Open-loop control systems

Open-loop systems are characterized by their lack of feedback. They rely solely on predetermined settings to regulate the input to the process without considering changes in the system's output. This means the control action is independent from the process output (Figure 1). While simpler, they often fall short in accuracy due to the absence of error correction mechanisms, making them sensitive to disturbances. This lack of adaptability makes them less suitable for complex, changing environments[23, p.11],[20, p.1, 2, 306],[19, p.15-16].

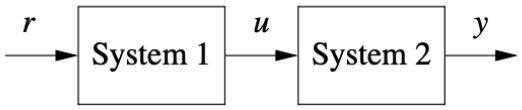


Figure 1: Open-loop control system. r = reference signal, u = control variable, y = process variable. *Figure from [20, p.1]*

As an illustration of an open-loop system, consider an electric space heater with a basic on/off switch which sets a specific power level. If you want to heat up the room you switch on the heater and set it's power. The room will start to heat up and reach a temperature which is determined by the heating power chosen. The temperature is thus controlled by initial settings without feedback and no further adjustments are made. This means the heater will continue to run on the set power level regardless of the actual room temperature. If there are changes in conditions e.g., someone opens a window, the heater does not compensate for the heat-loss. There is no information fed back to the heater to adjust and maintain a constant set temperature[19, p.15-16].

1.3.2 Closed-loop control systems

Unlike open-loop systems, closed-loop or feedback control systems integrate the system's actual output (i.e., the feedback signal) into the control process. By measuring the output and feeding it back to the controller, the system dynamically adjusts the input based on the difference between the desired set point and the measured output (error signal)(Figure 2). This continuous process allows the system to minimize control errors and adapt to disturbances and process variations. This ensures that the output closely aligns with the intended outcome, maintaining stability[19, p.15-16],[18, p.2],[21, p.60],[20, p.2, 3, 17, 18],[23, p.11-12].

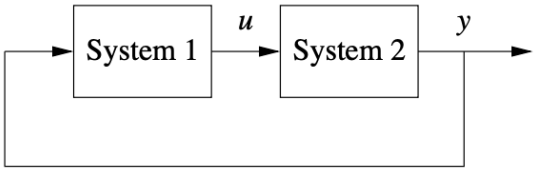


Figure 2: Closed-loop control system. u = control variable, y = process variable. *Figure from [20, p.1]*

A simple control loop consists of the following key elements:

1. The control unit which computes the control action using a predefined control strategy. It is the "brain" of the control loop. The control unit determines the appropriate action to maintain or change the process variable to meet the desired condition. It includes two main functions, the comparison element and the control law. The comparison element compares the actual value of the process variable (measured by the sensor) with the desired value or set point. The difference between these two is the error signal which tells you how far the system is from the desired state. The control law is a predefined strategy or algorithm that the control unit uses to compute the control action based on the error signal. It dictates how to correct the deviation. There exist different strategies that can be as simple as On-Off control and PID control or more complex types, like fuzzy logic control, model predictive control, adaptive control, neuronal network control and many others[24],[25],[26],[27],[28],[29],[30],[31],[32],[33],[34].

2. the actuator or final control element which produces a change in the process with the aim to correct the controlled condition. It translates the control action from the control unit to a physical force or movement, for example radiators in a home heating system.

3. the process or plant which is the system being controlled. It undergoes the desired change or maintains a certain condition as directed by the control action. For example in the context of a heating system, the process is the temperature regulation of a room.

4. the process sensor which measures the current state of the system. It provides real-time data about the process variable. In a home heating system, this could be the thermostat's temperature sensor, which continuously measures the room-temperature.
[19, p.18-19]

Control systems can be represented as a block diagram, where each part is illustrated as a single box with arrows denoting the causal relation between inputs and outputs(Figure 3)[21, p.5],[19, p.14-19],[20, p.4, 44],[18, p.2].

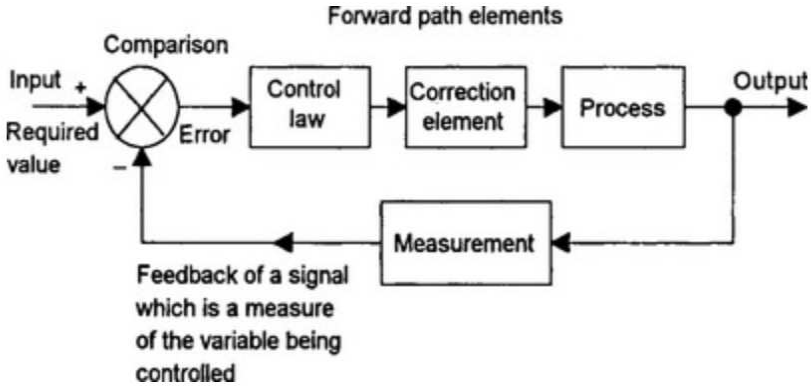


Figure 3: Basic elements of a closed loop control system. *Figure from [19, p.18]*

The process sensor continuously monitors the system by measuring it's current state, known as the process variable (PV), denoted as y. It reflects the real-time status of the process. The PV is then fed into the control unit which compares the PV with the desired state or set point (SP) of the system, denoted as y_{sp}. Any deviation of the process from the desired state generates an error signal $e = y_{sp} - y$. This error determines the necessary corrective action. Based on the error signal, the control law calculates the control action, denoted as u(Figure 4). This law defines the strategy or algorithm used to reduce the error. The actuator then applies the control action to the plant. This influences its behavior or state, resulting in a change in the output or PV. The resulting change in the PV is measured and then fed back as input into the control unit, closing the loop [18, p.2],[21, p.5, 60],[19, p.17-20],[20, p.4, 23, 24],[23, p.11-12].

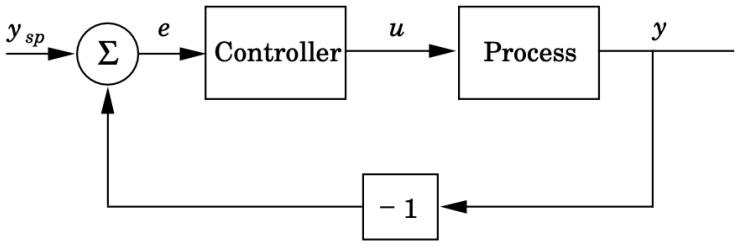


Figure 4: Block diagram of a simple feedback system. *Figure from [21, p.6]*

To illustrate a closed-loop control system, consider the space heater example from before with some modifications (Figure 5). Now, the heater has a built-in thermostat that is set to maintain the room at a constant temperature. In this system, the desired temperature is the set point (y_{sp}) that you want the room to be, say 25°C. The thermostat continuously measures the room's current temperature, the process variable PV (y), which might read 20°C. The actual and desired temperatures are compared and the difference, the error signal (e) is calculated. In this case the error would be 5°C, indicating the room is too cold. To compensate for this error, a control action (u) is applied to the system, which could be an instruction to the heater to increase the power output until the desired temperature is reached. In this case, there is feedback, information being fed back from the output to modify the input to the system. This feedback makes the system robust to disturbances such as if a window is opened and there is a significant heat loss. Then these changes in temperature are measured, fed back, and the output is modified accordingly to match the actual to the required values[19, p.16-18].

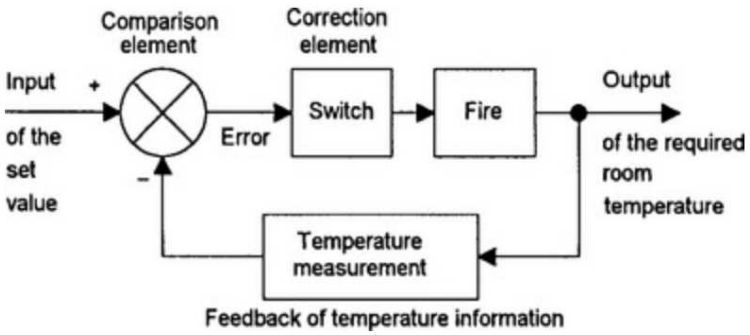


Figure 5: Closed-loop system with temperature feedback. *Figure from [19, p.17]*

The type of the control law used determines what action to take in response to an error signal. It can significantly influence the system's behavior and it's ability to achieve and maintain the set point [19, p.18]. This corrective action based can be implemented in many different ways[21, p.60].

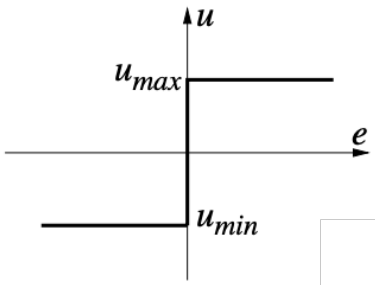
1.4 On-Off Control

The simplest and one of the most adopted form of control law is the on-off control, and can be described mathematically as follows:

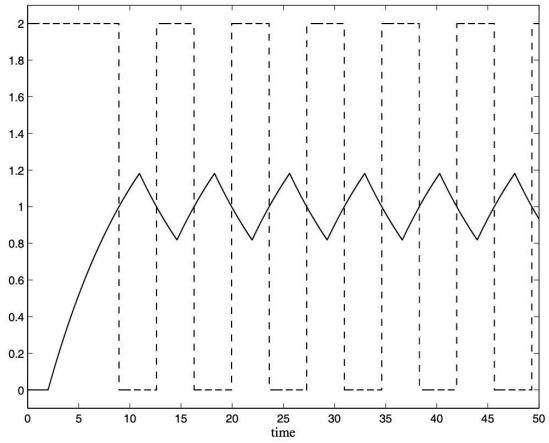
$$u = \begin{cases} u_{\text{max}} & \text{if } e > 0 \\ u_{\text{min}} & \text{if } e < 0 \end{cases} \tag{1}$$

where,
e is the control error,
u is the actuation command
Equation from [20, p.23].

This control law indicates that maximum corrective action is always used, where the control variable can assume one of two values either fully on, u_{max}, or fully off u_{min}, depending on the control error sign. This means, the control variable is set to its maximum value when the error is positive, where the process variable is below the set point, and its minimum value (zero or any other baseline level) when the error is negative, where the process variable is above the set point(Figure 6a). It is simple and there are no parameters to choose or adjust. Due to the discontinuous nature of the control action and the inherent time lags in system response, the process variable tends to oscillate around the set point(Figure 6b)[21, p.60-61]. This strategy works well in systems with large capacity, e.g. a central heating systems heating a large air volume. In such systems, changes in output result in slow changes in the process variable. Its simplicity makes the on-off controller a popular choice for basic control needs, specifically in scenarios when no tight performance is required[20, p.23-24],[18, p.3-4],[19, p.145-147],[23, p.93].



(a) On-Off controller characteristics.



(b) Example On-Off controller application. *solid line = process variable; dashed line = control variable*

Figure 6: On-Off Control. *Figure from [18, p.4]*

Let's consider a central heating system which has an on-off control strategy implemented(Figure 7). The system is set to maintain the temperature of the room at 20°C. Due to the on-off law, the heating system can be turned either fully on (maximum heating) or fully off (no heating). The system's components are as any other closed-loop system. The thermostat (sensor and controller) is measuring the current room temperature (process variable) and deciding whether to turn the heating system on or off based on the error (difference from the set point). The heater (actuator) is generating heat when turned on and stops when turned off. The room is the plant being controlled. The control law could operate as follows: if the room temperature drops below 20°C, say 19.5°C, the thermostat detects a positive control error (+0.5°C), since the actual temperature is lower than the set point). Following the on-off control law, because the error is positive, the thermostat responds by turning the heating system on to its maximum output capacity (u_{max}). The heater stays on until the room temperature reaches or slightly exceeds 20°C, say 20.5°C. Now, with the temperature above the set point, the control error becomes negative (-0.5°C) causing the heating system to turn off (u_{min}). As the room cools down over time, the temperature will eventually drop below 20°C again[19, p.145-147],[23, p.93].

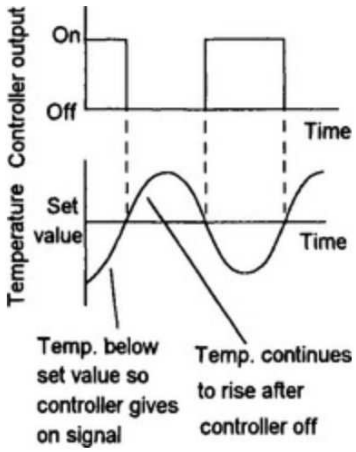


Figure 7: On-Off Temperature Control. *Figure from [19, p.146].*

1.5 PID Control

The PID controller or three-term controller is another control algorithm that is widely adopted in industrial control systems, particularly in the field of automatic mechanical, chemical and electrical process control. In process control more than 95% of control loops are of PID type. There are several key reasons for its broad application: Its intuitiveness, its relative simplicity in design and analysis combined with satisfactory performance for a wide range of processes[18, preface, p.1],[35, preface],[23, p.69],[21, p.1],[36]. Proportional-Integral-Derivative (PID) control is an essential concept in control theory. PID controllers use feedback to determine the necessary adjustments to maintain the desired system output and stability despite of disturbances and variations of process dynamics[20, p.23-25]. Implementing a PID control law involves applying properly the sum of three types of control action. These are a proportional action, an integral action and a derivative action. Each type responds to a different aspects of the feedback signal: The present error (proportional), the accumulation of past errors (integral), and the "prediction" of future errors (derivative). It represents a method to regulate systems effectively and responsively by adjusting control inputs based on the system's current state, historical performance, and anticipated future trends(Figure 8)[19, p.145-152],[18, p.2, 3, 19],[21, p.1].

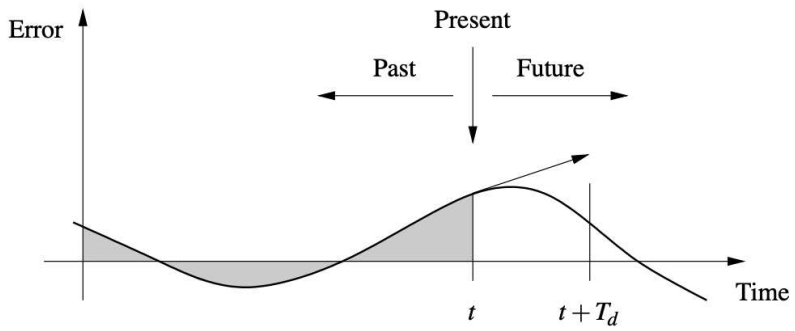


Figure 8: Action of a PID controller. At time t the proportional term depends on the instantaneous value of the error. *shaded area* = The integral term of the feedback, based on the integral of the error up to time t. The derivative term as an estimate of the growth or decay of the error over time by looking at the rate of change of the error. T_d = approximate amount of time in which the future error is estimated. *Figure from [20, p.25]*

The control action (u) is determined by these three actions, dictated by their proportional, integral, and derivative gains (K_p, K_i *or* τ_i, K_d *or* τ_d). They balance the need for quick responsiveness, error correction over time and predictive adjustment, thereby aligning the PV with the SP [23],[35],[20], [18],[19],[21, p.64].

The control action in a PID control system can be mathematically represented by the equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} = K_p \left(e + \frac{1}{\tau_i} \int_0^t e(\tau)d\tau + \tau_d \frac{de}{dt} \right). \quad (2)$$

where,
$u(t)$ represents the control action at time t,
K_p is the proportional gain,
K_i is the integral gain,
K_d is the derivative gain,
$e(t)$ is the error at time t, the difference between the set point and the actual process variable,
$\int_0^t e(\tau)d\tau$ is the integral of the error over time,
$\frac{de(t)}{dt}$ is the derivative of the error, or its rate of change,
τ_i is the integral time constant,
τ_d is the derivative time constant
Equation from [23, p.69],[20, p.293]

The time constants τ_i and τ_d are sometimes used instead of the integral and derivative gains (K_i and K_d). They determine the "weight" or significance given to past and future errors over a specified time frame. The integral time constant τ_i determines how quickly the integral action responds to the accumulated error. It sets the time frame over which the error is summed. The controller's integral action is stronger and more immediate with a smaller τ_i (inverse scale of the integral gain $\frac{K_i}{\tau_i}$) as it implies that a significant correction is applied over a shorter period. The derivative time constant τ_d defines the time frame for considering changes in the error. It determines how fast the controller reacts to the change in error. The derivative action is stronger with a larger τ_d (directly scale derivative gain $K_d * \tau_d$) [20, p.293-296],[22, p.151-154],[23, p.76-84].

Let's take a closer look at each control action of the PID controller.

1.5.1 Proportional control action

In a proportional (P) controller, the control output is directly proportional to the error signal. The error signal is defined as the difference between the reference signal (desired set point) and the output signal (the measured process variable of the feedback signal)[18, p.2],[35, p.1],[23, p.69],[19, p.150],[36]. Mathematically, this can be represented as:

$$e(t) = y_{sp} - y \quad (3)$$

where,
$e(t)$ is the error signal at time t,
y_{sp} is the reference signal,
y is the output signal,
Equation from [20, p.35],[21, p.11],[35, p.18],[22, p.122].

The P-controller amplifies the error signal by a constant factor, known as the Proportional gain K_p. K_p scales the error to generate an appropriate control action. Its a tuning parameter that adjusts the magnitude or strength of the controller's response to an error[18, p.3-4]. The choice of K_p significantly impacts the system's performance, including its stability, speed of response, and accuracy. With a properly selected K_p,

the system responds more quickly to errors, reducing the time it takes to reach the desired set point[23, p.70]. However, too high a K_p can cause the system to become unstable, leading to oscillations, while too low a K_p may result in a sluggish response (Figure 9)[18, p.16-17],[21, p.61-67],[36]. The controller's output is calculated as:

$$u(t) = K_p \times e(t) \tag{4}$$

where,
$u(t)$ is the control action at time t,
K_p is the proportional gain,
$e(t)$ is the error at time t
Equation from [35, p.1],[19, p.150],[18, p.3].

A proportional controller's response is directly proportional to the error. The nature of a P-controller (the control action corresponds to the error size) has the advantage of providing a small control variable when the control error is small and therefore avoid excessive or over-reactive control efforts (unlike On-Off control)[20, p.67, 36]. This leads to a smoother system operation under normal conditions. Further, a larger error induces a stronger response, facilitating an immediate and robust reaction to deviations from the set point[22, p.122]. By increasing the proportional gain, the bandwidth of the system is increased which leads to a faster, more responsive system[18, p.16, 33]. However, increasing K_p or system responsiveness to minimise the error leads to more frequent oscillations in the output signal and decreases the systems damping effect (reduce amplitude of oscillations over time), resulting in a higher overshoot (the extent to which the system exceeds the desired set point before settling). When just using the proportional mode of control minimising the error means the system becomes more oscillatory with increased overshoot[20, p.294]. The more responsive the system, typically, the greater the overshoot(Figure 9)[35, p.3, 7][p.age7],[19, p.141, 152],[21, p.67, 34],[36].
The tendency to overshoot also depends on the dynamics of the process being controlled. In systems with certain types of inertia or delay, even a correctly scaled proportional control action can lead to overshoot because the system itself doesn't respond instantaneously to the controller's action[21, p.113-120].

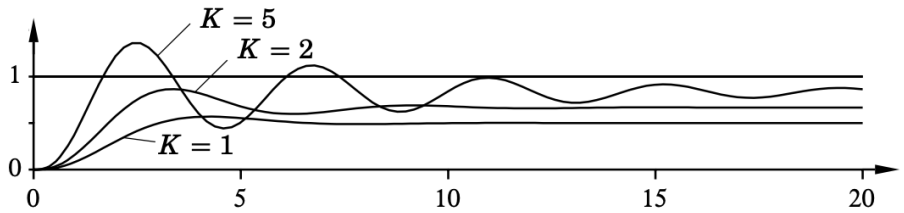


Figure 9: Closed-loop system with proportional control. Set point $y_{sp} = 1$; process output y for different values of controller gain K_p *Figure from [21, p.66]*

These two behaviors, small control for small errors and overshoot, occur under different conditions. Small control actions for small errors help in fine-tuning and maintaining system stability under normal operating conditions. Overshoot, on the other hand, is more prevalent in situations with larger error or rapid set point changes, where the proportional response can be more aggressive. The balance between these two aspects - avoiding overreaction to small errors while managing the risk of overshoot for large errors - needs to be considered when tuning a P-controller[23, p.70, 85],[19, p.141]. This is why in many practical applications, proportional control is often combined with integral and/or derivative control to overcome these limitations, to achieve both, precise small-error adjustment and effective handling of larger errors or dynamic changes.

Steady-state error
With pure proportional action, there is generally a control error in steady state[21, p.67]. The steady-state error is defined as the persistent difference between the set point and the process variable at steady state, a condition where the system has settled and no longer changes over time[19, p.141],[20, p.36],[35, p.132],[22, p.111-130],[21, p.127]. Despite the effectiveness of a P-controller in reducing the error, a significant limitation is its inability to completely eliminate it. The inherent characteristic of a P-controller (the control action is proportional to the error size) leads to decreasing response of the controller near the set point. In a typical control system, as its output approaches the set point, the error and consequently the controller's output becomes smaller, leading to insufficient response to bring the error to zero. The steady-state error can be reduced by increasing the gain. However, to achieve zero steady-state error, the gain would have to approach infinity[23, p.70-71]. The system under P-control might reach an equilibrium where the controller's output is constant, but is often insufficient to fully correct the deviation from the set point, leaving a non-zero steady-state error[20, p.294],[35, p.2],[36].
The inability of a pure P-controller to not fully correct the steady-state error to zero is especially apparent under conditions of constant disturbances, system inertia, fraction or rapid changes in set point values. The issue of steady-state error is further increased when integrating different systems.

Steady-state error in integrating systems with P controllers
Integrating dynamic systems are a type of control system where the output is the integral of the input over time. In other words, the system's response is based on the accumulation of the input over time. This type of system inherently sums or integrates its inputs, leading to a continuous increase or decrease in the system's output if there is any sustained, non-zero input[21, p.21, 209–212]. A P-controller, which only responds to the present magnitude of the error, may not be able to fully correct this type of error. This is particularly true with constant load disturbances, which are external influences that continuously affect the system and may continually introduce or sustain errors[18, p.4, 15]. The P-controller will adjust the output based on the current error, but since the system integrates over time, the error itself may also accumulate, preventing the system from reaching the desired set point precisely. Consequently, in an integrating system under constant disturbances, a P-controller alone often results in a non-zero steady-state error. Again, the system may reach a sort of equilibrium where the controller's output is constant, but it's not sufficient to fully correct the error and stabilize the system at the desired set point[21, p.12],[19, p.140],[18, p.15].
Whether in general systems with dynamic variations or specific integrating systems, both scenarios illustrate the fundamental limitation of a P controller in fully correcting steady-state error under certain conditions, whether due to the accumulating nature of an integrating system or the diminishing controller response as the output approaches the set point. These are not separate issues but rather two different manifestations of the same limitation. To overcome this challenge and achieve more precise and stable control, the proportional controller is combined with additional control action. Integral control not only helps in reducing steady-state errors by accumulating and responding to error over time, but it also contributes to a more balanced and stable system response[18, p.15-19].

1.5.2 Integral control action

The fundamental action of an integral (I) controller is to integrate the error signal over time. The control signal generated by an integral controller at any point in time is the cumulative sum (integral) of the error signal up to that instant[18, p.5]. The primary purpose of integral control is to eliminate steady-state error[20, p.24],[21, p.67, 127],[19, p.139-140],[35, p.5],[23, p.69-72],[22, p.124],[36].

The control signal $u(t)$ from an integral controller can be expressed as:

$$u(t) = K_i \int_0^t e(\tau)\, d\tau \tag{5}$$

where,
$u(t)$ is the control action at time t,
K_i is the integral Gain,
$e(\tau)$ is the error at time τ,
$\int_0^t e(\tau)\, d\tau$ is the accumulation of the error over time from 0 up to the current time t
Equation from [18, p.5],[20, p.24].

The integral gain K_i is the tuning parameter that adjusts the strength of the integral action to an error. Integral control action is designed to respond not just to the magnitude of the error, as in proportional control, but to the duration of the error. It integrates the error over time, producing a cumulative response, particularly effective in addressing sustained errors[18, p.5]. By continuously accumulating the error, the integral controller applies an increasing (or decreasing) action until the error is reduced to zero, continuously working to bring the system to the desired set point[21, p.67, 127],[19, p.140-141]. Due to these characteristics and effects on the system response, when tuned correctly, the integral action can improve system stability[18, p.16]. While integral control is effective in reducing steady-state error, it can also introduce challenges like overshoot and increased oscillations around the set point, particularly if the integral gain K_i is set too high and the controller reacts too sensitive to accumulated errors(Figure 10)[23, p.71, 85],[35, p.7].
In practical terms this can reduce the efficiency and precision of the process being controlled[20, p.296].

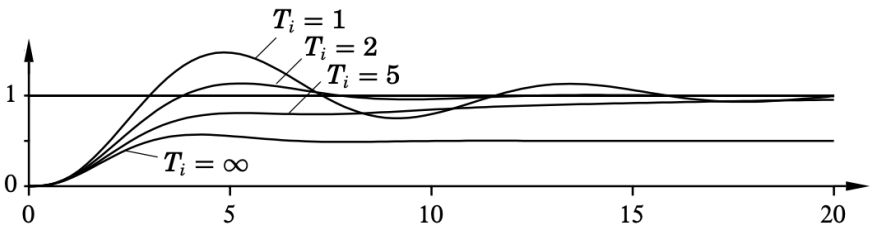


Figure 10: Closed-loop system with proportional and integral control. Proportional gain $K_p = 1$; set point $y_{sp} = 1$ and process output y for different values of integral time T_i *Figure from [21, p.68].*

A common issue and one of the most well-known sources of performance reduction when utilizing integral control is the integral windup phenomenon[18, p.35]. It occurs when the controller output saturates due to the actuators operational maximal (or minimal) limits, such as the maximum speed of a motor or a valve being fully open or close. This can happen even if the set point is not yet reached and the error signal is non-zero[20, p.225],[22, p.129]. If a controller with integrating action is used, the error will continue to be accumulated. When this happens the feedback loop is broken and the actuator will remain at its limit independently of the process output. Essentially, the controller's output can no longer influence the process. As a result, the integral term may become very large or, it "winds up". The control signal will remain saturated even if the process variable starts moving toward the set point. Once the error decreases, the accumulated error in the integral component can be so large that it takes a significant time for the system to return to normal operation. This results in long periods of overshoot, and in a sluggish response to changes in set point(Figure 11)[21, p.80-81],[20, p.225, 306–307],[18, p.5],[23, p.85-86].

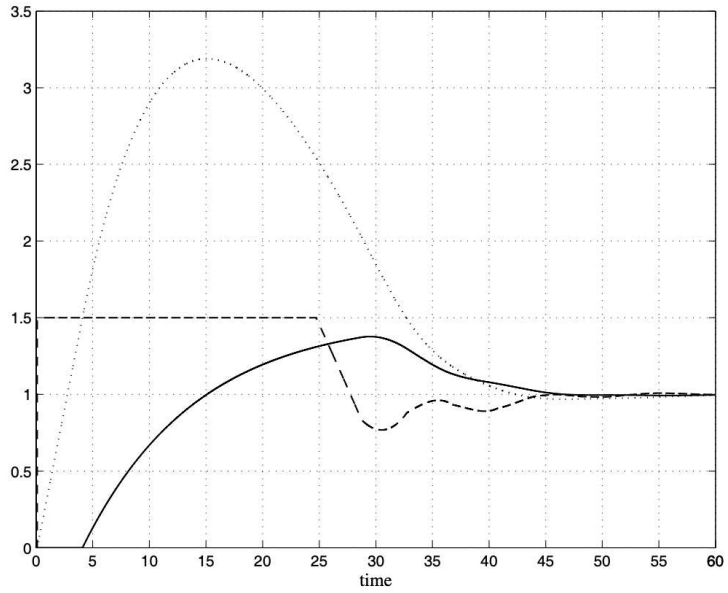


Figure 11: Integrator Windup. set point step response of the integral windup phenomenon. *solid line* = process output; *dashed line* = process input; *dotted line* = integral term *Figure from [18, p.36].*

Consider a heating system controlled by a thermostat set to maintain a room at 25°C. Integral windup can occur when there is a sudden drop in temperature (e.g., due to a window being opened in winter). To compensate for the lost heat, the system's response is to activate the heating at maximum capacity to return to the set point temperature of 25°C. The heating system has a finite capacity. This means, the actuator (heating element) cannot provide heating beyond a certain level due to inherent physical constraints (e.g., the heater's maximum power output). As the heater is already operating at its maximum, any additional heat required to counteract heat loss cannot be produced. This leads to a sustained error between the actual and desired temperature. The integral component accumulates this error over time. Even as the room temperature nears the set point, the controller's output remains at its maximum due to the large accumulated integral value. This leads to a delay in the system's response to changing conditions. The large error causes the heater to continue operating at full capacity even after the room temperature reaches the desired set point. The excessive heating action leads to overshooting the set point, making the room too warm. It takes a while for the accumulated error to dissipate ("un-wind") and the system to correct itself and stabilize[18, p.35-37].

1.5.3 Derivative control action

The derivative (D) controller operates based on the rate of change of the error rather than the error itself. The primary function of the derivative controller is to react to the rate at which the error signal is changing[19, p.150],[23, p.73]. The control output of a derivative controller can be expressed as:

$$u(t) = K_d \frac{de(t)}{dt} \tag{6}$$

where,
$u(t)$ is the control action at time t,
K_d is the Derivative Gain,
$e(t)$ is the error at time t,
$\frac{de(t)}{dt}$ is the rate of change of the error at time t
Equation from [18, p.6]

The derivative gain (K_d) is the tuning parameter that adjusts the strength of the derivative action to an error. Since the derivative component calculates the rate of change of the error, it reflects the instantaneous trend of the error. By applying a constant (K_d) to this rate, the controller effectively adjusts its output in anticipation of the error's direction of change. This allows the controller to take corrective actions in response to incorrect trends of the error signal[20, p.24, 296],[18, p.6]. One of the primary benefits of the derivative control is its contribution to the system's stability. By anticipating future changes in the error, it can initiate corrective actions early, which helps in stabilizing the system more quickly[21, p.69-70],[22, p.125]. This anticipatory action can lead to more effective control, especially in rapidly changing systems[21, p.111]. If the error is increasing rapidly, the derivative controller will act to apply a strong correcting force, and vice versa, thus anticipating and counteracting the error's trend before it becomes larger(Figure 12). This increased responsiveness (or increased bandwidth) is important in dynamic environments where the system needs to quickly adjust control actions in response to changes[18, p.6, 10],[19, p.150-152].

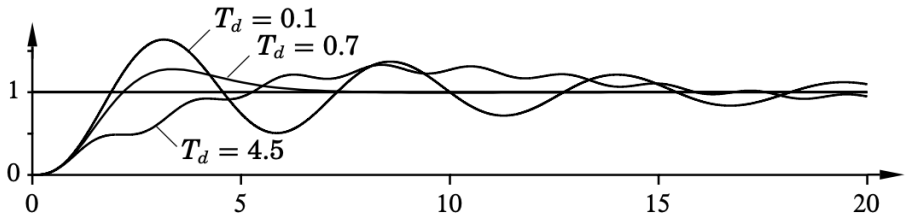


Figure 12: Closed-loop system with proportional, integral and derivative control. proportional gain $K_p = 3$; integral gain $T_i = 2$. Set point $y_{sp} = 1$ and process output y for different values of derivative time T_d. *Figure from [21, p.70]*

The Taylor series expansion is a mathematical method used to approximate the value of a function at a point based on its values and derivatives at another point. In the context of control systems, it's used to estimate the future behavior of the error signal. Consider the control error $e(t)$ at a current time t. The Taylor series expansion can be used to approximate the value of this error at a future time $t + T_d$ (where T_d is a small time interval)[21, p.69],[20, p.24].

A simple prediction is given by the linear extrapolation:

$$e(t + T_d) \approx e(t) + T_d \frac{de(t)}{dt} \tag{7}$$

where,
$e(t)$ is the error at time t,
$\frac{de(t)}{dt}$ is the rate of change of the error at time t,
$T_d \frac{de(t)}{dt}$ is the anticipated change in the error over the time interval T_d
Equation from [20, p.24],[21, p.69],[18, p.6]

The control signal is thus proportional to an estimate of the control error at time T_d. It predicts how the error will evolve shortly based on its current rate of change, particularly it predicts the error T_d time units ahead(Figure 13)[20, p.24, 297],[21, p.69].

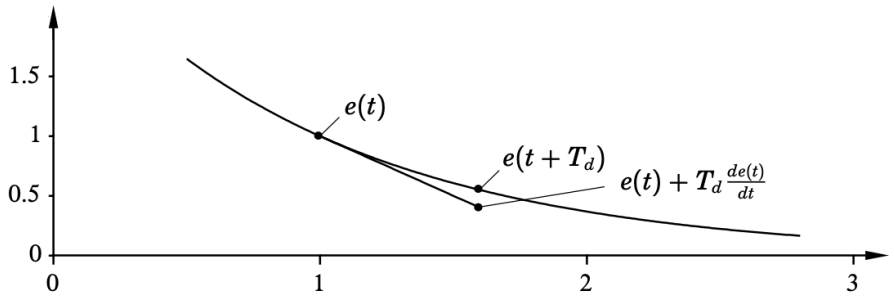


Figure 13: Interpretation of derivative action as *predictive* control - Tailor series expansion *prediction* by extrapolating the error by the tangent to the error curve (linear extrapolation). *Figure from [21, p.69]*

The derivative action can improve the control performance:

- Reduced overshoot: By adding damping, the derivative controller helps in reducing the overshoot that might occur in response to a change in the set point or a disturbance.
- Decreased settling time: The derivative action also improves the system's response by reducing the settling time, which is the time it takes for the system to stabilize at the new set point. In fact, the derivative action in a control system provides a phase lead, meaning the system's output starts responding before the input affects the system. This increases the system's bandwidth, making it quicker to adjust to changes in the set-point. A faster system response means that the process variable can more quickly reach and settle at a new set point, reducing overshoot and settling time.
  [19, p.152],[23, p.85],[20, p.24, 297],[21, p.111],[18, p.15-16].

However, it also has drawbacks because of which it is not commonly used in practice[21, p.2],[18, p.6-19],[21, p.1, 109],[35, p.5]. The derivative action in control systems inherently amplifies high-frequency signals (it is sensitive to high-frequency dynamics). While this is beneficial for enhancing the system's responsiveness to rapid changes, it simultaneously introduces the challenge of measurement noise amplification. Specifically, because an ideal derivative action inherently possesses high gain for high-frequency signals, any noise present in the measurements, which typically has high-frequency components, will result in large variations in the control signal. This sensitivity means that the controller might react not only to genuine changes in the system's behavior but also to random, high-frequency fluctuations caused by noise, and potentially leads to irregular control actions [20, p.308],[35, p.4],[18, p.15-16]. In real-world systems, measurement noise is

often present. It represents random disturbances that interfere with the accurate detection of process variables by sensors. These disturbances can alter the true signal being measured. Essentially, measurement noise distorts the actual signal, making it challenging to distinguish the real behavior of the process being controlled[21, p.54-55]. It can arise from environmental factors such as electromagnetic interference or from within the sensor itself including electronic noise in circuits[1, p.74],[19, p.72]. This noise can be represented as a fluctuating or oscillating signal, typically of high frequency. For example, a sinusoidal noise signal can be represented as

$$n(t) = A\sin(\omega t)$$

where A is the amplitude and ω is the frequency of the noise. When this noise signal passes through a derivative controller (with a derivative gain K_d), the output due to the noise becomes

$$u(t) = AK_d\omega\cos(\omega t)$$

This shows that the control action (output) is directly proportional to the frequency ω of the noise. Therefore, the higher the frequency of the noise, the greater the amplification in the control signal[18, p.9],[21, p.76]. In practice, this means that derivative control amplifies high-frequency measurement noise resulting in a "noisy" control variable signal, which can be erratic and unpredictable. In addition, in systems where the process dynamics have significant (apparent) dead time (a delay between when a control action is taken and when its effect is observed in the process variable) the derivative term may not contribute significantly to performance improvement. Dead-time makes the derivative term's ability to predict and correct future behavior less effective, as the process response itself is delayed relative to the control action[21, p.112]. Due to these challenges, most practical control systems use very small D or set its gain to zero (PI controller)[21, p.109, 1],[18, p.15-19],[35, p.5].

1.5.4 Step input and step response

The step response describes the relationship between an input that changes from zero to a constant value abruptly, a change referred to as a step input, and the corresponding output. The step response provides insights into the performance of a dynamical system. It characterises how a system reacts over time to changes in inputs (step changes). A unit step input is a tool for analyzing a system's dynamic properties. Mathematically, it is defined as:

$$u(t) = \begin{cases} 0, & \text{if } t < 0, \\ 1, & \text{if } t \geq 0. \end{cases} \tag{8}$$

Equation from [20, p.150].

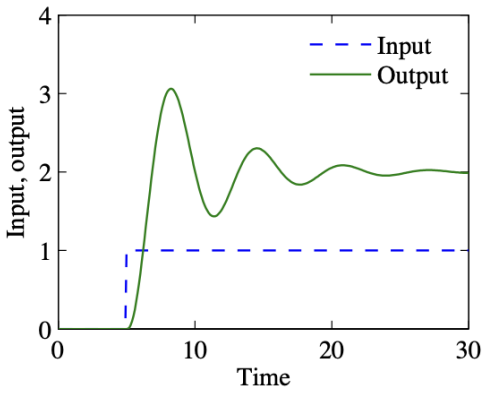


Figure 14: Step input (*blue line*) and corresponding step response (*green line*)
Figure from [20, p.31]

This abrupt change from 0 to 1 serves as a standardized input to examine a system's transient and steady-state behaviors, in other words, how the system transitions from its initial state to new equilibrium (Figure 14)[20, p.30-47],[21, p.8-13],[19, p.82-96]. A step response is typically characterized by several key parameters:

- Steady-state value (y_{ss}): The final output value of the system when it has settled and no longer changes over time.
- Rise Time (T_r): This is the time it takes for the system's response to rise to a certain percentage of the steady-state value, often from 10% to 90%. It indicates how quickly the system reacts to a change in input (system's responsiveness to changes).
- Overshoot (M_p): Defined as the maximum amount by which the initial response exceeds the steady-state, or final value. Future values do not overshoot the final value by this initial response. The overshoot is often expressed as a percentage. It is a measure of the system's stability and damping characteristics.
- Settling time (T_s): The amount of time required for the system's response to remain within a certain range (commonly 2% to 5%) of its final or steady-state value. Settling time reflects the duration needed for the transients to damp out (how quickly the system's output stabilizes) and return to equilibrium, where the amplitude of the oscillation should fall to be less than 2% or 5% of y_{ss} (Figure 15) [20, p.148-151],[21, p.126-127],[19, p.98-102],[23, p.45-48].

A system's step response can be experimentally determined, which is particularly useful for systems where theoretical models are complex or contain to many unknown variables. By applying a step input (changing the control variable rapidly to a new constant value) and observing the output, one can infer different aspects of system performance such as response speed (via rise time) and stability (via overshoot and settling time)[20, p.31-47],[21, p.8-11]. In practice, the step response can be utilized for PID controller tuning, where the controller's parameters are adjusted based on the response characteristics to achieve desired performance[21, p.11-13].

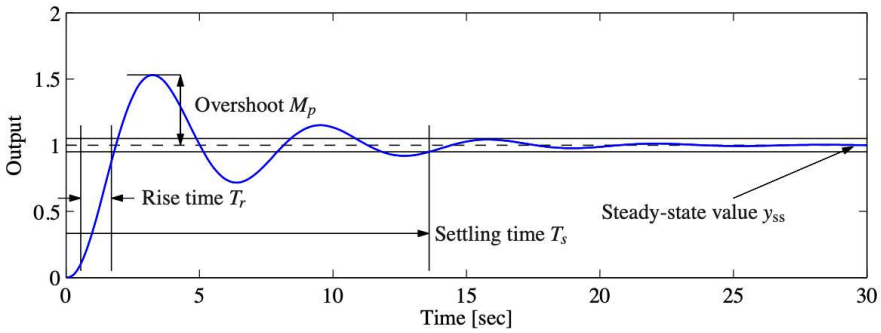
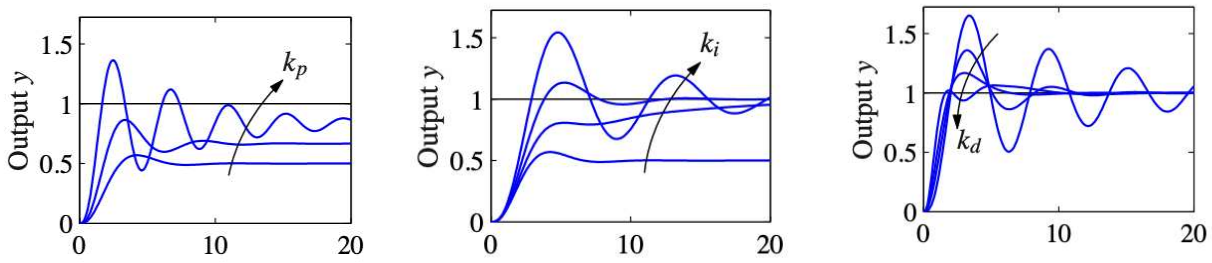


Figure 15: Sample Step Response. *Figure from [20, p.151].*

1.5.5 Tuning PID controllers

Tuning PID controllers is a fundamental aspect of control system design. By optimizing the controller's parameters - proportional gain (K_p), integral time constant (τ_i), and derivative time constant (τ_d) - it ensures that the system meets specific control requirements, such as effectively following set points, responding to load disturbances, minimizing control efforts (thus reducing operational costs and wear on actuators), and maintain robustness under varying conditions. For successful tuning, it is important to understand the role of each parameter. One of the major strengths of PID controllers is that their parameters have a clear practical meaning and their effects on the system can be observed[22, p.151-158]. Increasing proportional gain

K_p enhances the system's responsiveness (increased bandwidth), allowing for quicker reactions to errors. However, this potentially increases oscillations in the system's output, which could lead to instability if K_p is set too high (Figure 16a). Adjusting the integral time constant τ_i affects how aggressively the controller reacts to accumulated error over time. A larger τ_i (indicating a reduction in the integral action's effect / smaller integral gain (K_i)) tends to slow the system's response but contributes to overall stability by minimizing fluctuations and reducing steady-state error(Figure 16b). Increasing the derivative time constant τ_d introduces a damping effect, which can help stabilize the system by counteracting rapid changes in the error. However, excessive τ_d can lead to an undesired increase in system sensitivity to higher frequency components of the error signal, potentially destabilizing the system(Figure 16c). The control performance depends on properly chosen values for the parameters. When selected incorrectly, the system performs poorly and its response may become sluggish or unstable[18, p.16-17],[21, p.5-6],[22, p.151-154],[20, p.302-303],[36].



(a) Proportional Control. (b) PI Control. (c) PID Control.

Figure 16: Response to step changes for a system with a proportional controller (a), PI controller (b) and PID controller (c). proportional control parameters. $K_p = 1, 2, 5$; PI control parameters: $K_p = 1$, $K_i = 0, 0.2, 0.5, 1$; PID control parameters: $K_p = 2.5$, $K_i = 1.5$, $K_d = 0, 1, 2, 4$.
Figure from [20, p.295]

Tuning a PID controller can be addressed in various ways. The trial-and-error method is a practical approach which involves manually adjusting PID parameters and observing the system's feedback response to set point changes, step inputs or load disturbances and iteratively refining the parameters to achieve the desired behavior. This approach is particularly valuable when dealing with complex systems where precise mathematical models are unavailable or when the system's dynamics are not well understood[18, p.16-17],[21, p.5-6],[22, p.151-154],[20, p.302-303]. The process of trial-and-error tuning involves the following steps:

1. Proportional (P) control: Initially, the focus is on the proportional gain (K_p) parameter. Integral and derivative actions are set to zero. A step input is applied to the system, and the proportional gain is adjusted gradually. Observations are made to assess how the system responds to changes in K_p. The value of K_p should be as large as possible, to keep the system responsive. If process output oscillates, the proportional gain is too high; if the process output shows an over-damped response (the process output shows a slow and sluggish response), the proportional gain is too small.

2. Integral (I) control: Once an appropriate proportional gain is determined, integral action K_i is introduced. The integral action is adjusted while keeping the proportional gain constant. This step helps eliminate steady-state errors and improve system's response to changes. For a positive set point change, if the process output oscillates and stays above the set point for extended periods, then the integral time τ_i is too small (that is the integral action is too strong), leading to persistent overshoot. Conversely, if the process output oscillates and stays under the set point for extended duration after a positive set point change, then the integral time τ_i is too large (that is the integral action is too weak), resulting in slow error correction.

3. If necessary, derivative action (K_d) is added to the controller. Similar to previous steps, the derivative gain is adjusted while keeping the proportional and integral gains constant. Derivative can help

dampen the system's response to reduce overshoot. High frequency oscillations with many peaks in the process output from the beginning to the steady-state following a set point change indicate that the derivative time constant τ_d is set too large, amplifying high-frequency measurement signals [22, p.151-154],[23, p.82-84].

Another approach are the Ziegler and Nichols tuning rules. It is a widely used empirical method for determining optimal parameter settings based on simple characterization of process dynamics. Developed by John G. Ziegler and Nathaniel B. Nichols in the 1940s([37]), this method simplifies the tuning process by using critical process responses to set the proportional (K_p), integral (τ_i), and derivative (τ_d) gains. In a closed-loop configuration with only proportional control enabled (integral and derivative actions are set to zero), K_p is gradually increased until the system exhibits a stable oscillation. The value of K_p at this point is known as the critical gain k_c, and the period of the oscillation is the critical period T_c. Ziegler and Nichols provided empirical formulas to calculate the PID parameters based on k_c and T_c(Figure 17)[22, p.157],[19, p.159],[20, p.302-303],[36].

Type	k_p	T_i	T_d
P	$0.5k_c$		
PI	$0.4k_c$	$0.8T_c$	
PID	$0.6k_c$	$0.5T_c$	$0.125T_c$

Figure 17: Ziegler-Nichols tuning rules. Control parameters in terms of *critical gain* k_c and *critical period* T_c for different types of PID control. *Figure from [20, p.303],[22, p.157],[21, p.137].*

These settings are designed to provide good starting points for most control systems. However, because they rely solely on the information contained in the critical k_c and T_c, they may not be optimal for all processes, particularly those with significant dead-time or under-damped behavior (oscillations that gradually decrease in amplitude over time before eventually stabilizing at a set point), where they might lead to aggressive control actions or excessive oscillations[36][18, p.33],[21, p.36].

1.5.6 Choice of the controller type

The selection of the controller type is a critical step in control system design. The objective is to achieve optimal performance with the simplest possible configuration. A proportional (P) controller is the simplest form of control, adjusting the output solely in direct proportion to the error. While it's straightforward in design, its main drawback is the potential for a non-zero steady-state error. This might be acceptable in system where precision is less critical or systems of inherent integral behavior. Adding derivative control to form a PD controller can further improve system's' responsiveness and stability, by increasing the system's bandwidth and accelerating its response to changes. By providing improved damping, the derivative action allows for an increase in proportional gain without compromising system stability. However, when elimination of steady-state error is necessary, a Proportional-Integral (PI) controller is the best choice. Due to the integral action, the system is able to achieve its set point without residual error. This makes the PI control ideal for systems that require higher levels of accuracy and where load disturbances occur frequently. The PI controller is the most widely used type of PID control in industrial context. This is due to their effectiveness in a broad range of applications but also due to the problems associated with derivative action, namely the amplification of measurement noise and the difficulty in selecting an appropriate value of the derivative time constant (derivative gain) [18, p.15-19],[21, p.111-112],[35, p.5-8],[19, p.152],[20, p.296]. In a PI controller, the output is the sum of the proportional and integral control actions.

The control output is expressed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau \tag{9}$$

where,
$e(t)$ is the error at time t,
$K_i \int_0^t e(\tau)d\tau$ is the integral of the error over time τ
Equation from [35, p.5].

1.6 Building a Temperature Control System (TCS)

1.6.1 Choice of the controller type for the TCS

For developing the Temperature Control System (TCS) for a microfluidic device, a PI controller was chosen over a PID controller for several reasons:

1. *Sensor noise*: The thermistor used as the temperature sensor introduced significant high-frequency measurement noise. Thermistors, while offering high sensitivity and accuracy for temperature measurements, are prone to noise, including thermal noise and the effect of self-heating[23, p.16-24]. The derivative action led to an amplification of this noise and therefore to an unstable system with excessive spikes and fluctuations in the control signal.

2. *System dynamics*: The PDMS layer's high thermal mass and low thermal conductivity results in a system with inherently slow dynamics[38],[39],[40]. Rapid changes in temperature are therefore both uncommon and undesirable. Derivative control is most advantageous in systems that exhibit high dynamics and require instant responses to rapid changes. The potential benefits of derivative action in enhancing system responsiveness were not relevant in this scenario. Instead, the risk of noise amplification and the subsequent instability outweighed any potential advantages.

3. *Lack of detailed system model*: The absence of a model that describes the thermal behavior of the PDMS layer was another issue. The complex system dynamics and uncertainties made it challenging to predict the system's response and effectively tune the derivative component of a PID controller.

Considering the initial control requirement for dynamic temperature control and maintaining specific set points, a PI controller was chosen. It ensures stable and prompt control, effectively handles steady-state errors, and responds well to changes in the set point while avoiding the potential risks of instability associated with the derivative component.

1.6.2 Integral windup in the TCS

Integral windup can occur when an actuator reaches its operational limits (saturation) and cannot provide additional action required by the integral component. In the TCS, the primary actuators were the Peltier elements, responsible for heating and cooling to regulate the temperature of the PDMS layer in the chip. Given the PDMS' low thermal conductivity[38],[39],[40] considerable effort is required form the Peltier elements to induce the necessary temperature changes. In the specific experiment conducted, the relevant set points were 15°C and 25°C. Due to the close proximity of the set points themselves as well as their close proximity to the ambient temperature (maintained at 20°C), the required heat output from the Peltier elements did not push them into saturation, reducing the risk of integral windup. It allowed the Peltier elements to operate within their capabilities. It is important to note however, that while the set point temperatures for this experiment were within the optimal operating range of the Peltier modules, experiments requiring more extreme temperatures might produce a risk of integral windup. This would necessitate the implementation of anti-windup strategies[21, p.80-86],[20, p.306-309],[18, p.35-43].

1.6.3 Tuning the PI controller

The Ziegler-Nichols (ZN) tuning method was not successful for the TCS. Increasing the proportional gain did not lead to the expected oscillations. This is likely due to the PDMS' low thermal conductivity [38],[39],[40] which delays heat transfer and results in a dampened response to temperature changes. This inherent damping effect of the slow heat transfer resulted in a gradual rather than oscillatory system behavior, even at high proportional gains. The ZN tuning method relies on inducing and measuring clear, sustained oscillations to determine optimal controller settings[22, p.157],[19, p.159],[20, p.302-303],[36], and therefore it was found to not be the optimal method for tuning the TCS. The absence of a mathematical model, due to the system's complex and uncertain dynamics, further complicated the tuning process. Consequently, the trial-and-error method was used, by following the steps described in section 1.5.5. This approach is a more intuitive and adaptable tuning process, better suited to the specific characteristics of the system.

2 Materials and Methods

2.1 Circuit

The system hardware components with their connections (Figure 18) and functions in the circuit are listed:

- USP12837 Thermistor Probe: provides temperature feedback from within the PDMS layer of the chip. It is connected to the analog inputs of the microcontroller (Teensy 4.1), which uses the temperature data to compute the control action and adjusts the PWM signal accordingly.
- Teensy 4.1 microcontroller: It is the main control unit. It computes the control action based on the control algorithm implemented and the temperature data from the thermistor. It sends PWM signals to the H-Bridge driver to control the Peltier elements actions accordingly.
- L298N Dual H-Bridge Driver: it receives the PWM signal from the microcontroller and controls the direction (cool or heat) and amount of current (cooling and heating intensity) flowing through the Peltier elements.
- Peltier elements: connected to the output terminal of the L298N. They are heating and cooling devices to change the temperature in the chip. They operate in response to the current received from the H-Bridge.
- 12V Power Supply: connected to the L289N to provide the necessary power for the Peltier elements.

The corresponding Block diagram of the TCS closed loop system is shown in Figure 19.

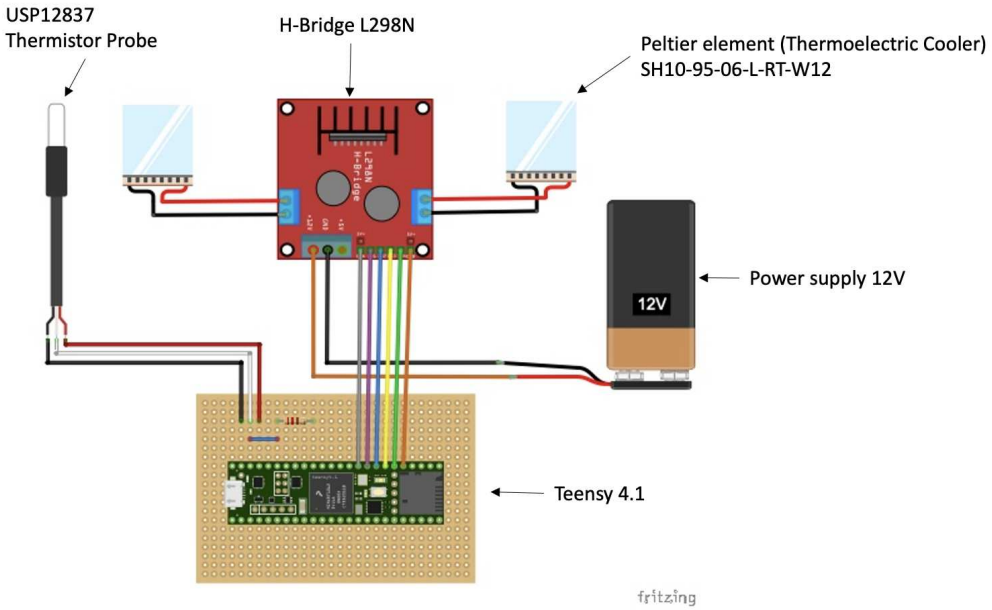


Figure 18: System Hardware Circuit.
made with Fritzing[41]

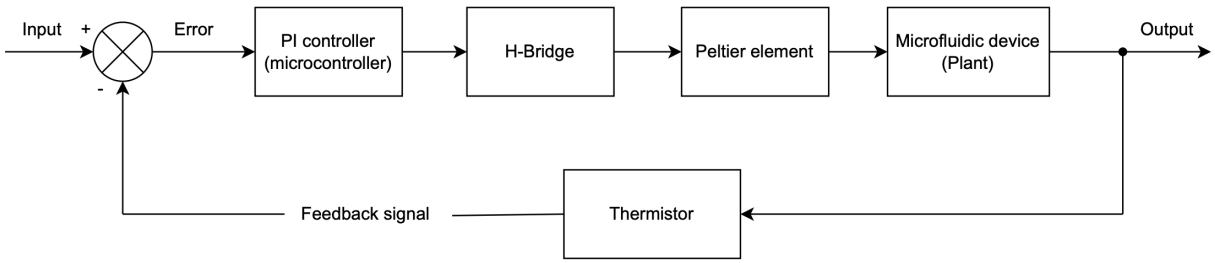


Figure 19: TCS Block diagram[19, p.7].

Temperature sensor

To measure the temperature within the PDMS layer of the microfluidic device, the USP12837 Thermistor Probe was used. It is a NTC (Negative Temperature Coefficient) thermistor. NTCs are temperature-sensitive semiconductors (e.g., metallic oxides) that exhibit a large change in electrical resistance when subjected to a small change in temperature, where resistance decreases as temperature increases. This sensitivity allows for a rapid response to temperature changes. To measure temperature, the NTC thermistor is part of a potential divider circuit, where a constant current is passed through the thermistor and a known, fixed resistor (1000Ω), and the resulting voltage drop across them is measured. This voltage drop indicates the thermistor's resistance at that temperature. NTC thermistors are often characterized by their resistance at a standard room temperature, usually 25°C [19, p.26-27],[23, p.23].

The USP12837 Thermistor Probe specifications:

- *Resistance* = 10,000Ω at 25°C: it provides a baseline for temperature measurements.
- *Temperature Coefficient* = -4.4%/°C at 25°C: it is a measure of sensitivity, indicating the rate at which the resistance decreases as the temperature increases.
- *Temperature Range*: -55 to +125°C; the upper and lower limits that can be measured.
- *Compact size* = 0.020" x 0.150" Dia or 0.5mm x 3.8mm: it's small dimensions make it ideal for applications where space is limited.
- *Remote measurement capability*: the long cable (12.0" $\pm$ 0.100" or ±30.48 cm $\pm$ 0.25cm) of the thermistor allows the temperature measurement at remote locations. The resistance of a long cable is negligible compared to the high resistance of the thermistor. This feature is particularly advantageous in setups where the probe needs to be placed in specific, hard-to-reach areas without interfering with other components or processes.
- *Resistance tolerance* = $\pm$ 1% at 25°C: is the precision with which the thermistor can measure temperature based on its resistance. This tolerance level means that the actual resistance of the thermistor can vary within a range of $\pm 1\%$ or $\pm 100\Omega$ from the nominal value of 10,000Ω, when measured at 25°C. In other words, the resistance could be as low as 9,900Ω or as high as 10,100Ω, and still be considered accurate within the specified tolerance. It directly affects how closely the thermistor's resistance (and thus the inferred temperature) matches the actual temperature.
- *R-T Curve* = J: is the resistance-temperature characteristic curve graphically illustrates the thermistor's resistance at various temperatures. For NTC thermistors, this curve will show a decrease in resistance

as temperature increases. However, this relationship is non-linear and the J type referrers to the particular shape of how the resistance changes over the operational temperature range (-55°C - 125°C) [42].

It is important to note that Thermistors can experience self-heating issues when current flows through them. This can lead to an increase in internal temperature, causing the resistance to decrease relative to its true value. As a result, there might be inaccuracies in temperature readings and fluctuating outputs [23, p.16-24]. As discussed in earlier sections, this type of noise might have contributed to an unstable control signal in the PID controller, particularly affecting the derivative component.

Actuators - Peltier elements

The Peltier modules, also known as thermoelectric cooler (TEC), serve as external heating and cooling elements. They are devices that use the Peltier effect to create a temperature difference across their two sides when an electrical current is passed through them. A typical Peltier element is composed of many thermocouples, which are pairs of n-type and p-type semiconductors, connected in series and layered between two ceramic plates[43]. N-type semiconductors are created by doping a silicon (or another pure semiconductor) with an element that has more valence electrons than the semiconductor itself (e.g., phosphorus with 5 valence electrons for silicon (4 valence electrons)). This adds extra electrons to the silicon lattice, which then become charge carriers with a negative charge, hence the name "N-type". These additional electrons enhance the material's conductivity since they can easily move and carry charge when a voltage is applied. In N-type materials, electrons are the majority charge carriers, and holes (positive charge carriers) are the minority carriers. Conversely, P-type semiconductors are produced by doping silicon with an element having fewer valence electrons (e.g., boron with 3 valence electrons). This creates "holes" in the silicon lattice where an electron is missing. These holes can "move" through the lattice when an electron jumps to fill a hole but leaves another hole in its original position. This "movement" of holes acts like positive charge carriers. In P-type materials, holes are the majority charge carries, and electrons are the minority charge carriers[44],[45]. The Peltier effect occurs at the junction of these two different types semiconductors. When current (electrons) flows from the n-type (electrons are in a higher energy state due to excess electrons) to the P-type semiconductor (electrons are in a lower energy state by filling holes (positive charge carrier) in the valence band). This transition releases energy in the form of heat, leading to a heating effect at the junction. Conversely, when electrons move from the p-type to the n-type semiconductor, they transition from lower to higher energy states. This requires the absorption of energy, which is drawn from the surroundings, resulting in a cooling effect at the junction. The absorbed energy facilitates the movement of electrons into the n-type material's conduction band. This electron movement creates a temperature differential across the module when a voltage is applied, with one side heating up (electrons moving from n-type to p-type) due to energy release and the other cooling down (electrons move from p-type to n-type) due to energy absorption from the environment. This process can be reversed, by reversing the direction of the electric current, the heating and cooling sides can be switched. By adjusting the direction and magnitude of the current through the Peltier module, the temperature of the microfluidic device can be controlled [16],[17].

H-Bridge

We use Peltier elements to heat or cool the chip. Now, we need to have control over how much they should heat or cool and which side is getting hot or cold. For this, we use a microcontroller, an H-bridge circuit, and the technique of Pulse Width Modulation (PWM). Microcontrollers use digital output pins (I/0 pins) to send control commands to other electrical devices. However, it can only handle and output low-power signals and can not directly manage the high current demands of Peltier elements. To bridge this gap, an H-bridge is used, it is a type of electronic circuit designed to handle high power levels. The microcontroller sends control signals to the H-bridge, dictating both the direction and the PWM duty cycle for magnitude control. The H-bridge then translates these low-power signals into high-power current control for the Peltier elements[46],[47]. By activating specific switches within the H-bridge, based on the microcontroller's signals, the direction of the current flowing through the Peltier elements can be reversed. Reversing the current flow allows to switch between heating to cooling modes, and thereby deciding which side of the Peltier module

becomes warm and which becomes cold. In addition, the microcontroller can modulate the magnitude of the current using PWM signals. Pulse Width Modulation (PWM) is a technique for controlling the average voltage delivered to an electronic device. It achieves this by rapidly switching the power on and off. By varying the ratio of the "on" time to the "off" time to the total period of the signal (the duty cycle), it is possible to control the average power delivered to the Peltier elements. The H-bridge responds to these PWM signals from the microcontroller, by adjusting the current flow accordingly. This means we have fine-tuned control over both, the heating and cooling intensity by varying the PWM output, and the change between heating and cooling modes by reversing the current direction[23, p.123-126],[48]. Specifically, in the TCS, the system's PWM frequency is set to 36,621 Hz, meaning the signal completes an on-and-off cycle 36,621 times per second. This high frequency enables smooth control of the output by reducing noise and allowing finer adjustments. With a 12-bit resolution, the duty cycle of the PWM signal can be adjusted in 2^{12}, or 4096, discrete steps. This high resolution means you can tune the duty cycle with great precision and consequently allow accurate temperature control. Here, however, the system's PWM signal is capped at 1500, despite the 4096 maximum value allowed by the resolution. This means the highest achievable duty cycle is significantly lower than 100%, specifically around 36.6% (1500/4096 * 100). This limitation helps to ensure that the heating element does not operate at full capacity. By keeping the duty cycle within this limit, we ensure the average current draw stays within the 2A limit of the power supply used in this setup, avoiding overloading the power supply or overheating the system.

Microcontroller - Teensy 4.1
A computer controlled system generally consists of a plant to be controlled, including sensors and actuators, and a microprocessor which acts as a central control device that stores control algorithms and computes the input to the plant. Practically, all control systems today are microcontroller based[23, p.12, 91]. This is due to fast data processing and fast control logic execution, which allows for real-time control of systems with minimal delay. In addition, they are small in size and very cost-efficient. Here, the microcontroller Teensy4.1 is used as the central control unit, to implement the PI control algorithm and for user interaction. A microcontroller is basically a computer on a chip. It is designed to perform specific tasks or control a specific system. It includes a processor (CPU), memory (both RAM and ROM/flash), and input/output (I/O) pins[49],[50]. The processor acts as the "brain" of the microcontroller. It processes data and executes instructions based on the stored program it runs. In this case, this program is the control algorithm, specifically, the PI controller. The input/output (I/O) ports allow the microcontroller to communicate with the external world. These pins can connect to sensors, actuators, or other electronic devices. This enables to receive input signals and send output commands, here changes in temperature from the thermistor and the control actions, respectively. Many microcontrollers include additional built-in function like timers, counters, ADC (analog-to-digital converters), and DAC (digital-to-analog converters). These features increase the microcontroller's ability to interact with other devices and control a wide range of systems[23, p.12, 91],[19, p.11-12],[35, p.113-114],[21, p.1, 93].

The role of the microcontroller (Teensy4.1) in a simplified version of the TCS's control loop:

- It continuously reads temperature data from the thermistor $y(t)$ to monitor the temperature.
- The analog (continuous) signal from the thermistor is converted into a digital (binary) form through an ADC (analog-to-digital converter).
- It compares the actual temperature to the desired temperature set point. This comparison generates the feedback error ($e(t)$).
- It executes the control logic. The implemented control algorithm computes the control signal ($u(t)$) based on the feedback error (K_p) and the history of the feedback error (K_i), to determine whether to heat or cool and how aggressively.
- Based on the output of the control logic, it sends commands to the actuators (H-bridge and Peltier

elements) to adjust the temperature of the plant (microfluidic device), moving it close to the desired state.

- It can interact with the user through a python interface to monitor system status, change parameters or temperature set points
[35, p.113-114],[21, p.1, 93].

For programming, the Teensy 4.1 is compatible with the Arduino Integrated Development Environment (IDE), requiring only the addition of the Teensyduino add-on. The programming language used is a variant of C++. The advantages of C++, particularly when working with the limited computing resources on a microcontroller, is its efficiency in terms of execution speed and system resource use[49],[50].

2.2 Arduino PI Controller Implementation

```
void loop() {
    // If user enters commands, i.e. "kp=0.3" or "Stop", the corresponding values
 are updated
    checkUserInput();

    double voltageSum = 0;
    for (int counter = 0; counter < 10; counter++) {
        voltageSum += readThermistorInput(thermistorPin);
    }
    double averageVoltage = voltageSum / 10;

    currentTemperature = convertVoltageToCelsius(averageVoltage);

    if(outputOn) myPID.Compute();

    // Adjust Peltier output according to computed PID output
    adjustPeltier(output);

    // Since system is sluggish, Arduino can cool down
    sleep(1000);
}
```

The given pseudo-code describes the implementation of the PI controller using the Arduino IDE. It calculates the PI control action output based on the temperature data it receives to adjust the Peltier elements performance. User input commands are checked for parameter adjustments such as PI settings and changing set points, or toggling the system's operational mode (on/off). For the temperature measurement, 10 voltage readings from the thermistor are collected, summed and the average voltage is calculated. The voltage value is then converted into degrees Celsius (C°)[51]. Thermistor readings can be quite noisy, so taking a moving average helps filter out these short-term fluctuations from the data and the actual temperature is represented more stable. Further, the smoothed data is more reliable for making control decisions where the system becomes more responsive to genuine temperature trends rather than transient changes. The PI controller computes the required adjustments to reduce the error between current and target temperature. The controller's output, the Peltier element's actions is adjusted accordingly. After this, the system sleeps for one second. The plant being controlled, the PDMS layer of the chip, does not respond instantaneously to changes in input. This slow reaction time, means that reading temperature too frequently would not provide significantly different information within short intervals. The one-second sleep allows the system to more fully respond to the last control action before taking the next measurement. This way each reading is meaningful and contributes to a more effective control. Since we use a PI algorithm in the TCS, there are no components that respond to rapid changes in the measured variable (temperature) which would require fast sampling rates. Therefore, such rapid control action adjustments are unnecessary. Minimizing the frequency of measurements and control adjustments also conserves resources such as electrical power and computational processing cycles. For implementing the PI controller, the PID library from Brett Beauregard[52] is used. The compute function of the PID library is described. The proportional control action is

calculated as the error and multiplied by the proportional gain. The integral control action is calculated by summing the error over time and then multiplying this sum by the integral gain. To calculate the derivative control action, the difference between the current input value and the previous input value is computed to find the rate of change, which is then multiplied by the derivative gain. The final output combines these three actions. The proportional and integral components are directly added together, while the derivative component (calculated as a negative because it is subtracted and therefore dampens the system's response) is based on the input change. The gains are previously set by the user. Specifically, in the TCS the derivative gain was set to zero. Lastly, the current input and time are updated for the next iteration[52].

```
bool Compute() {
    long now = millis();
    long timeChange = (now - lastTime);

    if(timeChange >= SampleTime) {
        double error = setpoint - currentInput;
        double inputChange = currentInput - lastInput;

        // Update output accumulator
        outputSum += ki * error;
        if(outputSum > outputMax)
            outputSum = outputMax;
        else if (outputSum < outputMin)
            outputSum = outputMin;

        // Compute output
        double output = kp * error;
        output += outputSum - kd * inputChange
        if(output > outputMax) output = outputMax;
        else if (output < outputMin) output = outputMin;

        lastInput = currentInput;
        lastTime = now;
        return true;
    }
    else return false;
}
```

With the control law implemented and having all the hardware components together, the TCS should also provide specific functionalities to communicate with the system, make it more straightforward and user-friendly.

1. Real-time temperature visualization: Observing real-time temperature changes over time helps to understand the current state of the system and how it behaves under different conditions.

2. Interactive control: The TCS needs to be able to send input commands to the Teensy to dynamically adjust temperature set points, control parameters, and output levels without stopping the system. This, together with real-time temperature plotting allows tuning the PI controller and evaluating system performance. A trial-and-error tuning approach is now possible where iterative improvements can be made by adjusting control settings in real-time and observing the outcomes. In addition, being able to change temperature set points and output levels on-the-fly and observe the instantaneous system response allows system performance analysis, such as conducting step response and benchmarking tests.

3. Data logging: Temperature data, the corresponding PWM output level and the duration of the measurement need to be saved in order to conduct post-operation analysis.

However, the Arduino IDE is not well suited for these kind of tasks. C++ code written in the Arduino IDE needs to be compiled and uploaded to the microcontroller as a static program. This means that any adjustments to parameters (like new temperature set points or PI tuning parameters (K_p and K_i)) require recompiling and uploading the code, which interrupts the system's operation. Without the ability to dynamically

adjust control parameters and simultaneously visualize the system's response to these changes, makes it tuning the PI controller difficult. In addition, the Arduino IDE does not support logging or saving data to a file on the computer. While it is possible to send the data to the serial monitor, saving this data for later analysis or visualization needs external software[50].

2.3 Python - User Interface (UI)

To overcome the limitations of the Arduino IDE the Teensy was integrated with Python. The integration supports real-time monitoring, communication and dynamic control, and appropriate data handling. These different tasks need to run simultaneously but at the same time should not interfere with each other so that the system stays responsive. Multiple threads allow running different tasks concurrently.

- Communication Thread: Dedicated to interact with the microcontroller. It runs continuously and exchanges input commands and data (temperature, control output values, timestamp, and duration of the current control state). Commands to the Teensy are queued for asynchronous operation where commands can be sent without interrupting other processes. It then writes the received data to an output file for logging and later analysis.
- User Interface (UI) Thread: It updates the graphical interface at regular intervals (every 200 ms) to display real-time temperature data received from the microcontroller. Specifically, the last 100 temperature readings, the output levels, current control settings and the duration of the current operation are plotted. Command inputs can be sent directly to the Teensy4.1 through the UI. These commands include setting the desired temperature (set point), tuning the PID by adjusting the control parameters (proportional, integral and derivative gains) to optimize systems response, and directly setting output levels delivered to the Peltier elements. The corresponding system responses can be observed in real-time. It also checks the status of the optional benchmarking thread, to adjust the UI based on whether benchmarking is active.
- Benchmarking Thread: It runs automated benchmarking tests by systematically testing the system's response to various settings. It does that by adjusting the temperature set point or control outputs over a specified range and time interval. This process is initiated through the UI.

```
# Main function managing threads
def main():
    arduino_thread = Thread(target=arduino_communication)
    arduino_thread.start()

    # Main loop for rendering UI, which calls animate in 200 ms intervals
    animation.mainloop()

    set_stop_event()
    arduino_thread.join()

    if benchmark_thread:
        benchmark_thread.set_stop_event()
        benchmark_thread.join()

# Animation Thread for updating and interacting with the UI
def animate(last_100_temperatures, current_output, duration):
    plot.update(last_100_temperatures)
    labels.update(last_100_temperatures, current_output, duration)

    check_thread_status(benchmark_thread)

# Arduino Communication Thread
def arduino_communication(stop_event, output_file, temperature_list, command_queue):
while not stop_event:
    if command in command_queue:
```

```
        send_arduino_command(command)

    arduino_input = read_from_arduino()

    current_temperature, current_output, timestamp, duration = parse_input(
    arduino_input)

    with open(output_file):
        write_to_file(current_temperature, current_output, timestamp, duration)

    if len(temperature_list) >= DISPLAY_LIMT:
        temperature_list.pop(0)
    temperature_list.append(temperature)

# Benchmarking Thread is started when UI button is clicked
def benchmarking_process(stop_event, setpoint, current_value, setpoint_type):
while not stop_event and not current_value == final_value:
    if setpoint_type == BENCHMARK:
        send_arduino_command(f"setpoint={value}")
    else:
        send_arduino_command(f"output={value}")

    current_value += step_size
```

2.4 Assembling the TCS onto a Spinning Disk Confocal Microscope Stage

The experimental setup involves mounting the system into an inverted confocal microscope, that is optimised for whole brain calcium imaging in *C. elegans* (microscope termed in lab jargon ZIM01). Two Peltier cells are arranged on either side of the microfluidic device, which allows for a uniform temperature distribution across the device. A thermal paste improves temperature conduction by filling air gaps which can act as thermal insulators[12],[11]. Aluminium heatsinks, attached to the Peltier cells, dissipate excess heat particularly when the system is in cooling mode. These heatsinks are integrated into an active cooling circuit that includes a circulating water bath. For the set up the LAUDA L100 thermoelectric circulation thermostat was used (LAUDA DR. R. WOBSER GMBH & CO. KG. LAUDA LOOP L 100 Thermoelectric circulator 100-240 V; 50/60 Hz Part Number: L000027, for more information go to LAUDA). It is a cooling and heating device designed for temperature control in laboratory applications[53]. By pumping chilled water through the heatsinks, it draws heat from the hot side of the Peltier module to prevent overheating. A custom-designed 3D-printed holder designed with OpenSCAD software [54] is used for placement of the system's components, specifically, the heatsinks, the Peltier elements, and the microfluidic device (Figure 20). This holder is placed on the microscope's Piezo stage, ensuring stability and accurate alignment of the temperature control elements with the microfluidic device. To manage unwanted heat absorption from the microscope's objective, a copper wire is coiled around it for active cooling. The copper wire is integrated into the same cooling circuitry that includes the Peltier module's heatsinks and the water-bath. The thermistor is placed within the PDMS layer in close proximity to the worm channel for accurate temperature monitoring. For this procedure, a 0.8mm steel wire was used to puncture a hole into the PDMS bottom layer, stopping just before reaching the worm channel. The thermistor was then be inserted into the pre-made hole using forceps. This direct temperature measurement allows for real-time temperature feedback, ensuring that the observed temperature accurately reflects the specimen's environment. All components of the temperature system are connected (H-Bridge, Peltier elements, micro-controller, thermistor, power supply)(Figure 21).

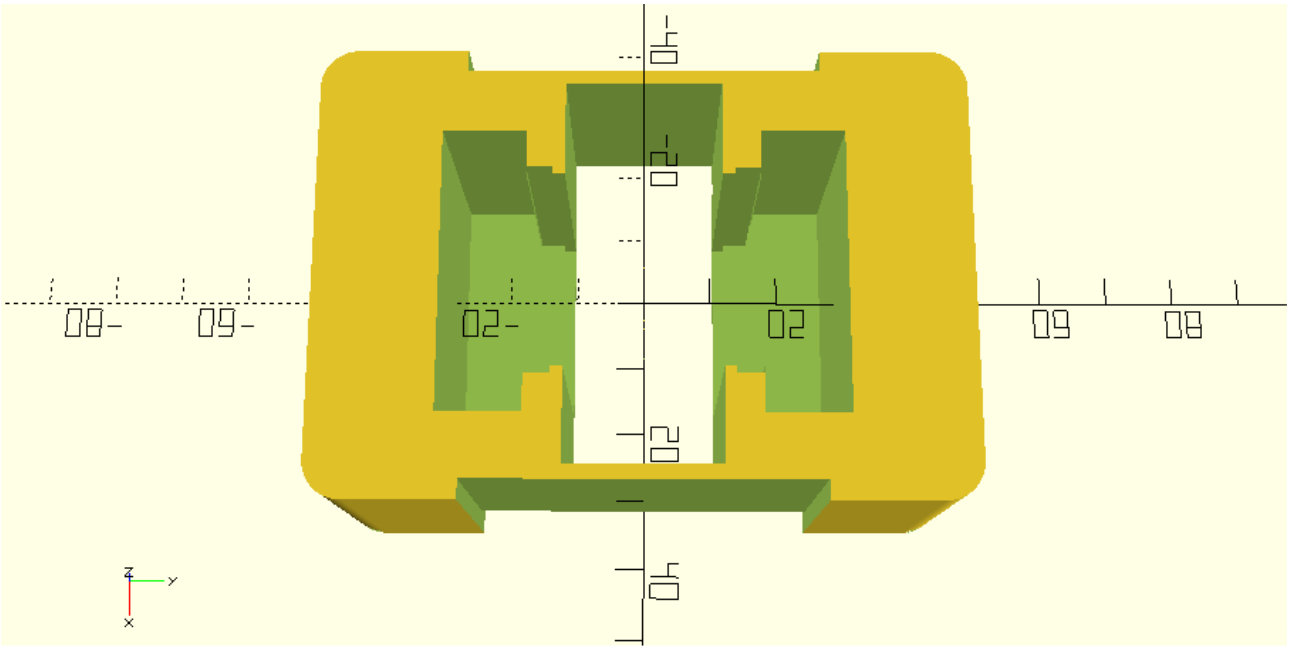
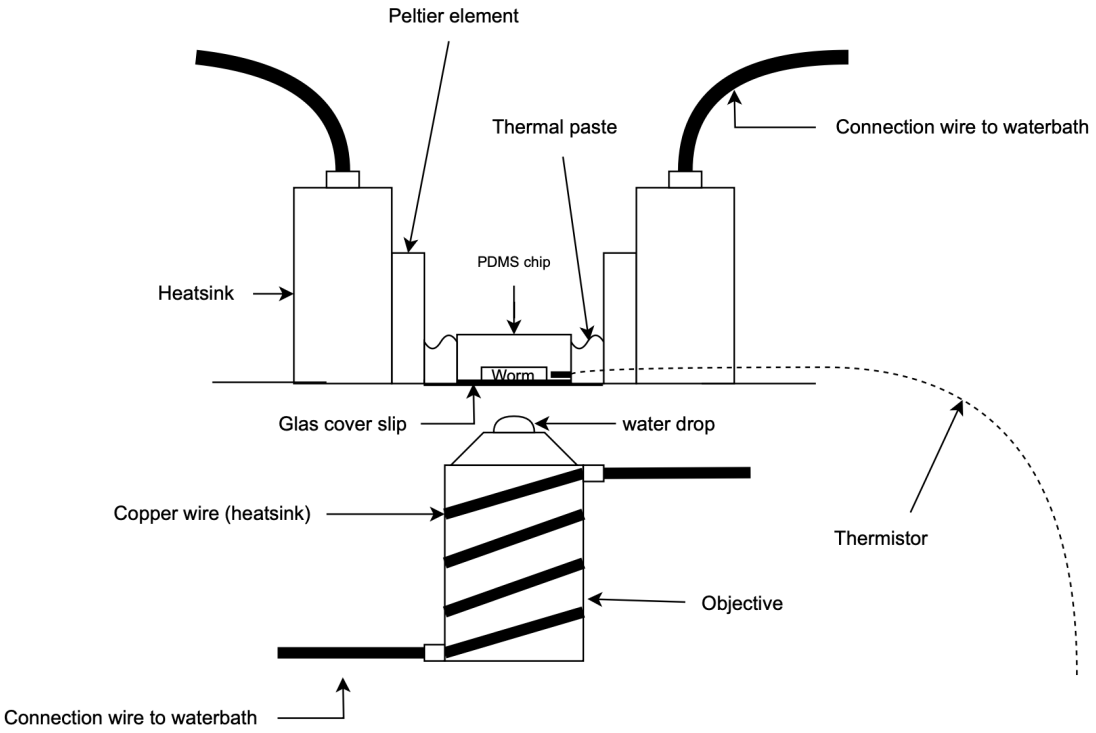
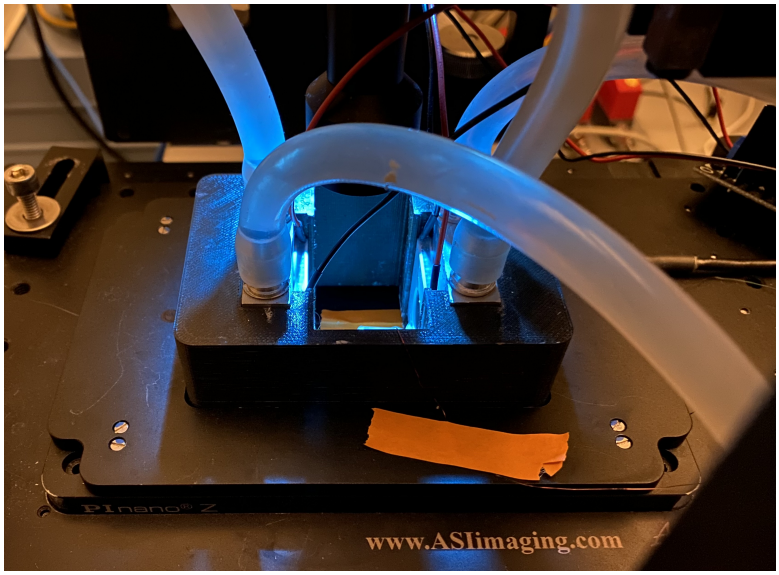


Figure 20: ZIM01 Holder. *3d-model design. made with OpenSCAD software[54].*



(a)



(b)

Figure 21: System setup in ZIM01.*(a) System schematics.(b) System mounted in ZIM01.*

2.5 *C. elegans* strains

Strain	Working Name	Genotype	Source
ZIM2423	Control line	mzmls 120, lite-1(ce314), mzmIs52;mzmIs120=[mzmEx1341=[mlc-2::HisCl(new)::rpl-3 3'UTR;Flp-17::scarlet]; zmIs52 = [mzmEx771 = [unc-31::NLSGCaMP6f cod opt intron]	Zimmerlab
ZIM2400	ts-mutant *cha-1* mutant	*cha-1*(md39) IV. MzmIs120 (mlc-2::HisCl(new):: rpl-3 3'UTR;Flp-17::scarlet] MzmIs52 (unc-31:: NLSGCaMP6fNLS) lite-1 (ce314)	Zimmerlab

Table 1: *C. elegans* strains.

2.6 Worm Maintenance

C. elegans strains were grown and maintained on nematode growth media (NGM) agar plates with E. coli OP50 as food source. Mutant as well as control lines were grown and maintained at 15°C except for the 2 hour acclimatization period at 25°C.

2.7 Whole-brain Calcium Imaging

Calcium imaging via GCaMP6f expression and worm immobilization
Whole-brain calcium recordings are performed in immobilized *C. elegans* control and *cha-1* mutant strains. Real-time visualization of neuronal activity was achieved through pan-neuronal expression of the nuclear localized calcium indicator GCaMP6f. It fluoresces upon calcium binding capturing subtle changes in intracellular calcium levels and serves as an indirect indicator of neuronal activity, described by Dana et. al. in 2019[55]. To immobilize the worms the histamine-HisCl1 system as described by Pokala et. al., 2014[56] was used. Histamine gated chloride channels are expressed in the worm's muscles under the control of the *mLc-2* promoter, leading to muscle relaxation and immobilization[56].

Imaging procedure and system setup
Cha-1 mutant worms are initially cultivated at the permissive temperature of 15°C. For imaging at the restrictive temperature (25°C), they are transferred to 25°C incubator for a two-hour acclimatization, with the final 30 minutes on histamine plate for immobilization. Worms with the most severe phenotypic manifestations of the *cha-1* mutation (coiled posture) where selected for imaging. This ensures observation of the most pronounced effects on neuronal activity. Control recordings involving mutants were kept and imaged at 15°C are prepared by transferring them on histamine plates 30 minutes prior imaging, without the 25°C incubation. This process is replicated exactly with control worms for both temperature conditions. Worms were picked onto an empty agar plate and washed with NGM mixed with 1M histamine. They were then loaded into a microfluidic chip as previously described [3],[9],[8]. The microfluidic device allows to constrain the worms in a standardized position for imaging. It consists of a Polydimethylsiloxane (PDMS) layer, and two channels, one for worm placement, the other for gas flow (21% O2 with 50 ml/min flow rate). The imaging setup comprised an inverted Zeiss Axio Observer Z1 confocal microscope, equipped with a Yokogawa CSU-X1 spinning disk and a Photometrics Evolve 512 EMCCD camera. A 40x water-immersion objective lens was utilized for acquiring high-resolution whole-brain activity data. The worms were illuminated using a 488nm

laser from Toptica Photonics, with a calibrated exposure time of 20 ms per frame. Imaging involved capturing a series of Z-stacks with 2μm intervals, covering 18-21 planes which results in an acquisition rate of 2.4 - 2.8 volumes per second. Prior to imaging, the TCS is pre-adjusted to the required experimental temperature, either 25°C or 15°C. This pre-regulation ensures that the PDMS layer reaches and stabilizes at the target temperature for the imaging session. Once the system has settled to the desired temperature, the selected worms are loaded for imaging.

2.8 Data Analysis

2.8.1 System Performance and step response parameter calculation

The system performance is assessed by analyzing its response to a set point change and is evaluated by the step response parameters rise time, overshoot and settling time.

Rise time
The rise time in a control system is the time it takes for the system's response to go from a certain lower percentage of the final value to a higher percentage, typically form 10% to 90% for a step response[20, p.148-151],[21, p.126-127],[19, p.98-102],[23, p.45-48]. In the context of the TCS if heating, the rise time is the time taken for the temperature to rise from 10% to 90% of its final value (set point), relative to the initial temperature. Conversely, for cooling (fall time), it's the time it takes for the temperature to fall from 90% to 10% of the final value (set point), relative to the initial temperature. The values corresponding to 10% and 90% of the way form the initial temperature to the set point are determined. Then, the timestamps at which the temperature first crosses these 10% and 90% thresholds are identified. The rise time is the difference between these to timestamps, indicating system response speed.

10% and 90% threshold values,
$T_{10\%} = T_{initial} + 0.1 \times (T_{setpoint} - T_{initial})$
$T_{90\%} = T_{initial} + 0.9 \times (T_{setpoint} - T_{initial})$

Here, $T_{10\%}$ is the initial temperature at the start of the observation period, $T_{90\%}$ is the target set point temperature.

Rise time,
$T_{rise} = t_{90\%} - t_{10\%}$

Here, $t_{10\%}$ is the timestamp when the temperature first crosses $T_{10\%}$, and $t_{90\%}$ timestamp when the temperature first crosses $T_{90\%}$.

Peak temperature and overshoot
The peak temperature is the highest (for heating processes) or lowest (for cooling processes) temperature value recorded during the control process. It represents the maximum deviation from the set point[20, p.148-151],[21, p.126-127],[19, p.98-102],[23, p.45-48], either in the positive direction (overshoot) or negative direction (undershoot), depending on the system's initial condition relative to the set point.

For heating,
$T_{peak} = max(T)$

For cooling,
$T_{peak} = min(T)$

Here, T_{peak} is the peak temperature observed in T, the set of all recorded temperatures. Overshoot refers to the extent to which the temperature exceeds (in heating) or falls below (in cooling) the set point. The

overshoot is calculated as the absolute difference between the peak temperature and the set point.

Overshoot in absolute numbers,
$OS = |T_{peak} - T_{setpoint}|$

Overshoot as a percentage,
$OS\% = \left|\frac{T_{peak} - T_{setpoint}}{T_{setpoint}}\right| \times 100$

Settling time
settling time is defined as the time required for the system's response (temperature) to remain within a certain range or tolerance around the desired set point after initially reaching it. This range is originally given as a percentage (2% to 5%) of the final value or set point[20, p.148-151],[21, p.126-127],[19, p.98-102],[23, p.45-48]. This can be a challenge in systems with varying set points like the TCS. When the tolerance range is defined as a percentage of the set point, the actual temperature range that determines "settled" can significantly vary with the set point value. For example, a 2% tolerance on a 15°C set point equals to a $\pm$0.3°C range, whereas the same 2% tolerance on a 25°C set point equals to a $\pm$0.5°C range. To address this, the tolerance range is defined in absolute numbers, particularly $\pm$0.5°C from the set point. This means, the temperature is considered settled if it remains within 0.5°C above or below the set point.

Tolerance range,
$T_{upper} = T_{setpoint} + \Delta T$
$T_{lower} = T_{setpoint} - \Delta T$

Here, T_{upper} and T_{lower} are the upper and lower bounds (based on the acceptable deviation from the set point), and ΔT is the tolerance value. The stabilization point is the earliest time at which the temperature (T) enters the tolerance range and remains within it for the remainder of the observation period. Find, the smallest time t_{stable} for which,

$T_{lower} \leq T(t) \leq T_{upper}, \forall t \geq t_{stable}$

Settling time,
$T_{settle} = t_{stable} - t_{start}$

Here, t_{start} is the timestamp at the start of the observation period.

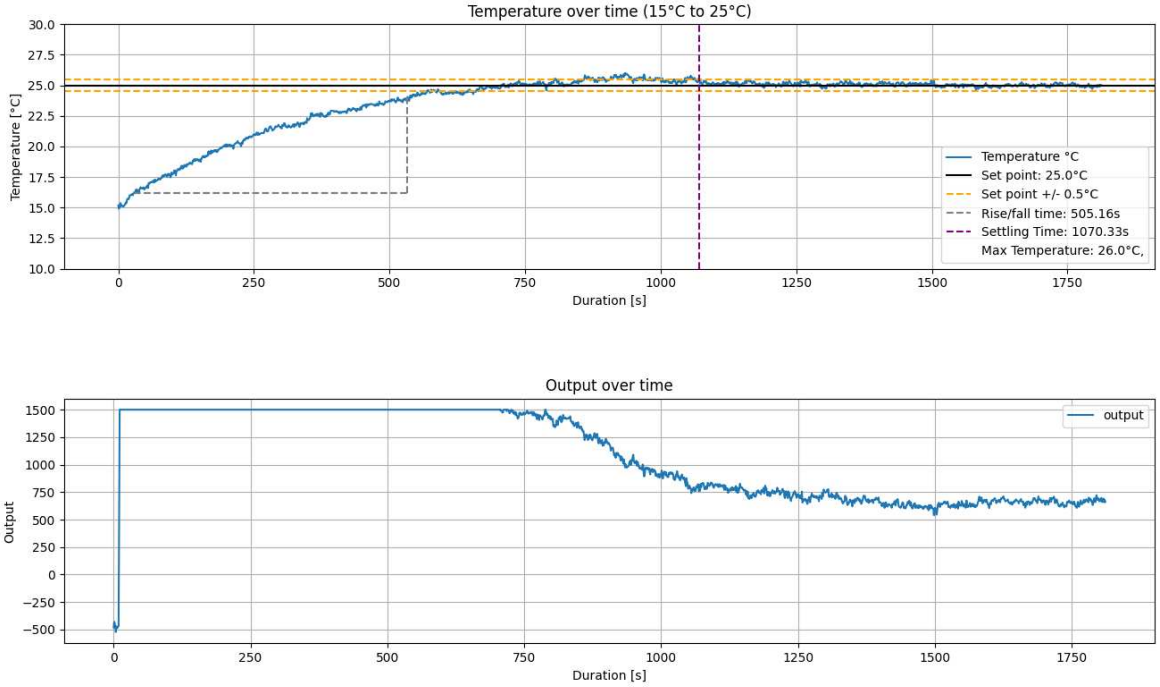
2.8.2 Neuron tracing and identification

Neuronal activity was tracked using MATLAB software to trace the intensity maxima of each volume over time as described by Kato et. al. (2015) and Uzel et. al. (2022) [3],[57]. This involves identifying the brightest points in each image volume, which represents active neurons, and tracking these points across the time series of images. Single-cell fluorescence intensity, the activity level of individual neurons at each time point (F) is calculated. After background subtraction the fluorescence changes ($\Delta F/F_0$) are calculated, with F_0 being the mean fluorescence intensity across the trial. All traces were corrected for bleaching as described by Kaplan et. al. (2020)[9]. Neurons are identified based on their specific activity patterns and anatomical locations.

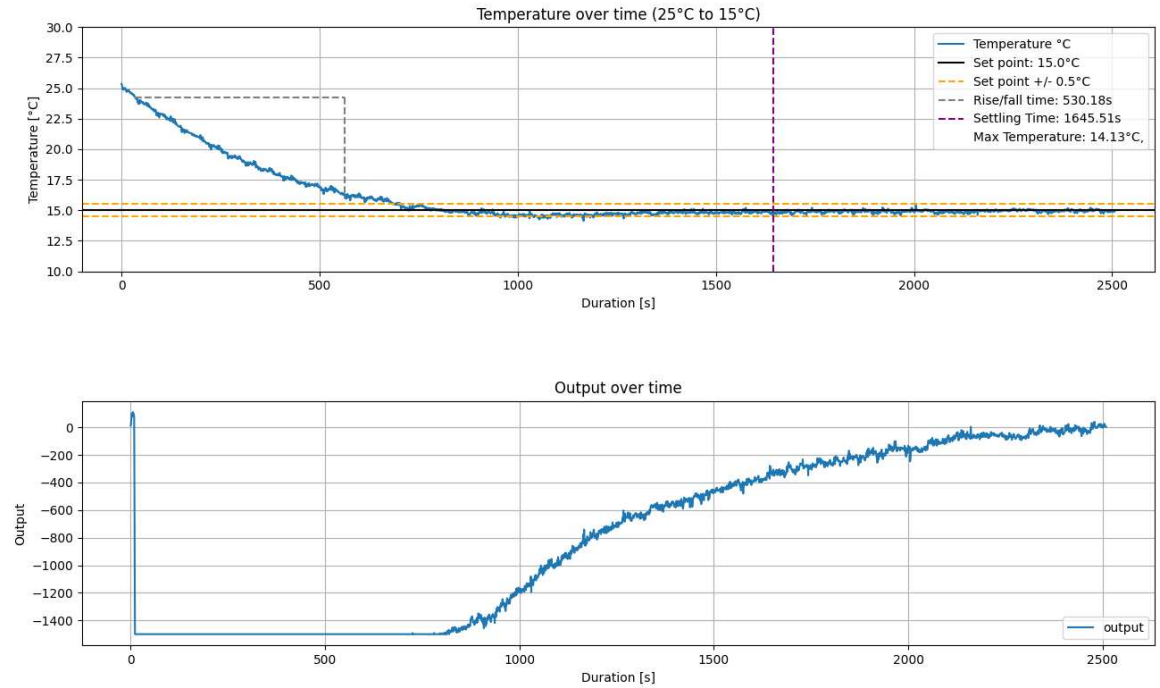
3 Results

3.1 TCS Performance

The performance of the TCS was evaluated through a series of step response experiments. A total of twelve runs under different conditions (heating and cooling) and under different start- and set point temperature (5°C and 10°C step size) with three trials each were conducted. The transitions for the heating trials were performed from 20°C to 25°C and from 15°C to 25°C, whereas cooling trials were performed from 20°C to 15°C and from 25°C to 15°C. The start- and set point temperatures were chosen according to the permissive (15°C) and restrictive (25°C) temperatures of the *cha-1* mutant worms as well as the room temperature (20°C). The step response behavior are used to assess the system's performance, by obtaining the performance parameters, rise/fall time, overshoot, peak time, and settling time. Specifically, the system was evaluated by its ability to reach and maintain a desired set point with minimal deviation. Figure 22 shows two typical step responses of the TCS upon a set point change of a 10°C step size, from 15°C to 25°C for heating, (Figure 22a) and from 25°C to 15°C for cooling (Figure 22b). The temperature over time is shown, with the blue line representing the actual temperature of the system as it responds to a step change in set point temperature as well as the corresponding system output behavior. The performance measures, rise/fall time (time from 10% to 90% of the final or steady state value), settling time (time when the system settles within a predefined tolerance range around the set point), and overshoot (max. Temperature) are shown. These two runs are representative for all 12 runs performed (supplementary Figures 29, 30, 31, 35, 36, 27, 28, 32, 33, 34), and show the standard form of the step response curve. It shows a steady temperature increase when heating and a steady decrease when cooling, indicated by the rise and fall times respectively. Following the rise/fall time, the system reaches it's peak time where it slightly overshoots before it settles back into the defined tolerance band around the set point, indicated by the settling time.



(a)



(b)

Figure 22: Set point Step Response. *Response dynamics of the TCS for a given step input.(a)Heating, step size of 10°C from 15°C to 25°C.(b)Cooling, step size of 10°C from 25°C to 15°C.*

On average, the system required an rise/fall time of 534s. A slight overshoot was observed in all trials with an average excess of 0.85°C above the set point. Notably, the overshoot in cooling conditions was consistently lower, with an average of 0.79°C. To reach the maximum temperature (max. overshoot) the system required an average peak time of 1143s. Settling times varied across runs but the system was generally able to stabilize the temperature within an average of 1457s. The high variability in parameters, indicates that the system performance is not consistent across different runs. This leads to the assumption that different condition (cooling or heating) or different categories (5°C or 10°C step size) might affect system performance (Table 2, Figure 23). The performance data for all individual runs is listed in supplementary Table 5.

Measurement Type	mean	std	SEM
Peak time	1142.96	364.07	105.10
Rise/fall time	534.43	214.5	61.92
Settling time	1457.48	408.00	117.78
Overshoot	0.85	0.13	0.04

Table 2: TCS Performance. *Descriptive statistics (n=12).*

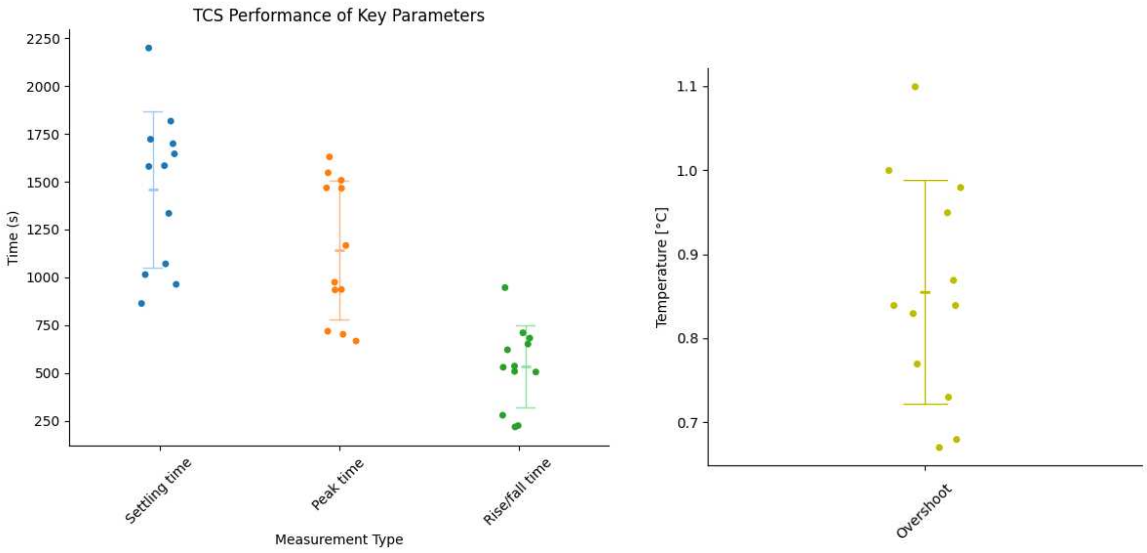


Figure 23: TCS Performance. *Performance Parameters, settling time, peak time, rise/fall time and overshoot for each run performed with the TCS. Each data point represents the parameter values measured for twelve runs (step response) in both, cooling and heating. Plots show mean ± Std.*

3.2 Comparison Heating and Cooling across Step Amplitudes

The robustness of the system was evaluated by comparing its performance across varying step sizes in both heating and cooling modes. Specifically, if the system's performance measures, rise/fall time, settling time, peak time, and overshoot vary with category (5°C or 10°C step size) or condition (heating or cooling).

Measurement Type	Condition	Category (step size [°C])	mean	std	count	sem
Settling time	Cooling	5 (20-15°C)	1581.90	617.73	3	356.65
		10 (25-15°C)	1519.80	163.93	3	94.65
	Heating	10 (15-25°C)	1536.17	406.22	3	234.53
		5 (20-25°C)	1192.06	445.07	3	256.96
Rise/fall time	Cooling	5 (20-15°C)	496.84	241.88	3	139.65
		10 (25-15°C)	524.84	14.76	3	8.52
	Heating	10 (15-25°C)	711.23	221.99	3	128.17
		5 (20-25°C)	404.79	265.87	3	153.50
Peak time	Cooling	5 (20-15°C)	1271.79	486.18	3	280.69
		10 (25-15°C)	1025.66	123.33	3	71.20
	Heating	10 (15-25°C)	1329.42	342.75	3	197.88
		5 (20-25°C)	944.97	451.08	3	260.43
Overshoot	Cooling	5 (20-15°C)	0.72	0.05	3	0.03
		10 (25-15°C)	0.85	0.02	3	0.01
	Heating	10 (15-25°C)	0.93	0.22	3	0.13
		5 (20-25°C)	0.92	0.08	3	0.05

Table 3: Heating and Cooling Performance. *Descriptive statistics for Cooling and Heating*

When comparing overshoot in both categories for the heating condition, they show a similar mean overshoot at 0.93°C for 10°C, and 0.92°C for the 5°C step size. The error bars for the heating condition generally show a wide range, particularly for the 10°C step size category, suggesting more variability in the overshoot during heating. The cooling conditions show less variability and a mean overshoot of 0.72°C for the 5°C, and a mean overshoot of 0.85°C for the 10°C step size. In general, the system tends to overshoot less when cooling (mean: 0.79°C) compared to heating (mean: 0.92°C) as well as when the step size is smaller (mean 10°C step size: 0.89°C; mean 5°C step size: 0.82°C)(Table 3, Figure 24a). The peak time shows noticeable variability within the heating condition, especially in the 10°C step size category, but overall, the system takes a similar amount of time to reach the peak temperature in both cooling (mean: 1,149s) and heating (mean: 1,137s). The system tends to be faster in 5°C step size in heating (mean 5°C step size: 945s; mean 10°C step size: 1,329s) but not in cooling (mean 5°C step size: 1,272s, mean 10°C step size: 1,026s)(Table 3, Figure 24b). Rise/fall times are generally shorter for cooling (mean: 511s) compared to heating (mean: 558s), and more consistent for cooling, particularly for the 5°C step size. The system tends to be faster in the 5°C step size category in both cooling (mean 5° step size: 497s; mean 10°C step size: 525s) and heating (mean 5°C step size: 405s; mean 10°C step size: 711s) conditions(Table 3, Figure 24c). The cooling condition tends to be slower in reaching settling (mean: 1,551s) compared to heating (mean: 1,364s). However, variability is again more pronounced in the heating conditions. The system tends to reach settling faster in the 5°C step size in heating (mean 5°C step size: 1,192s; mean 10°C step size: 1,536s) but not in cooling (mean 5°C step size 1,582s, mean 10°C step size: 1,520s)(Table 3, Figure 24d). The differences in performance parameters tested, however, are not significant when compared between conditions and categories(Figure 24, Table 4).

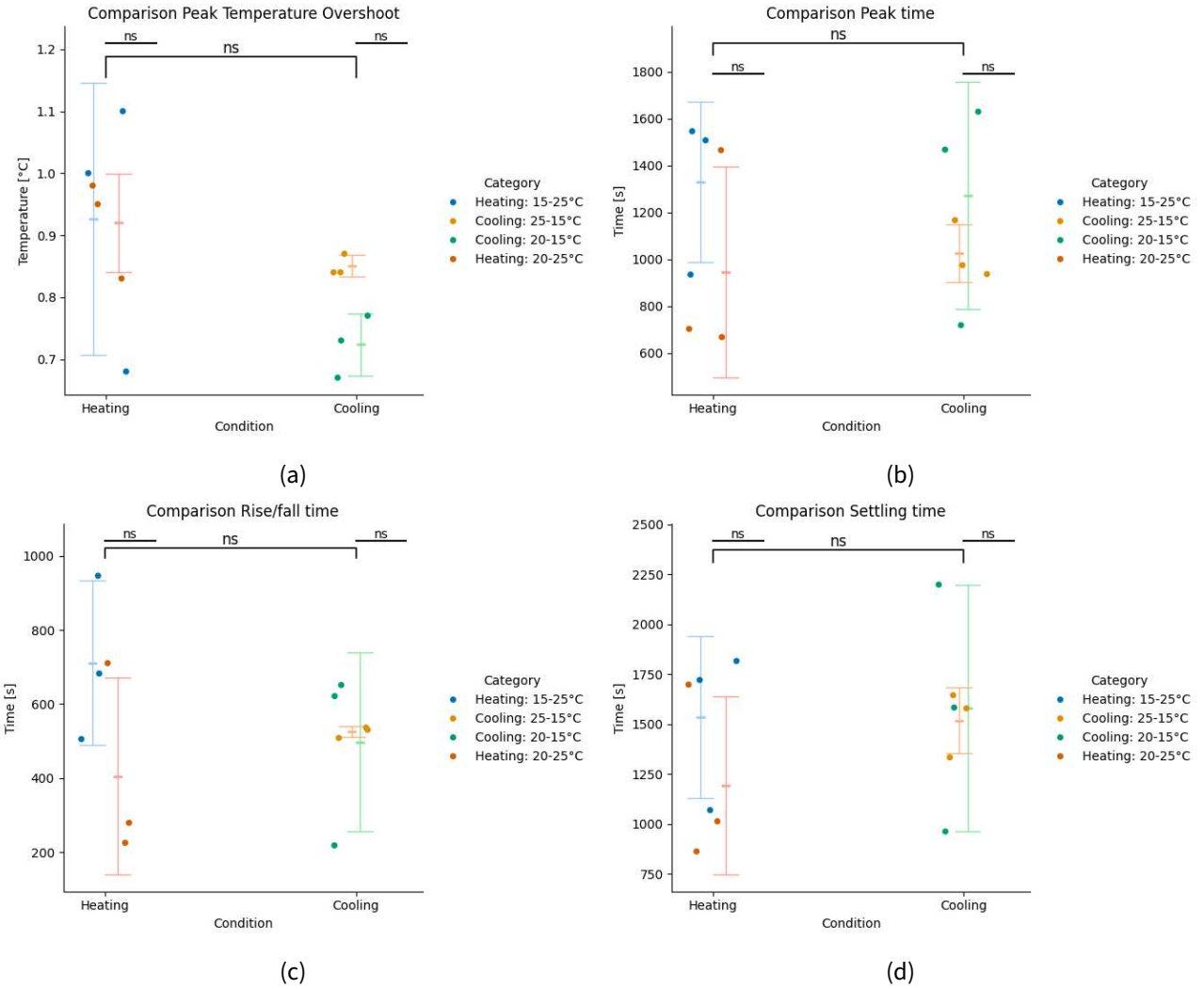


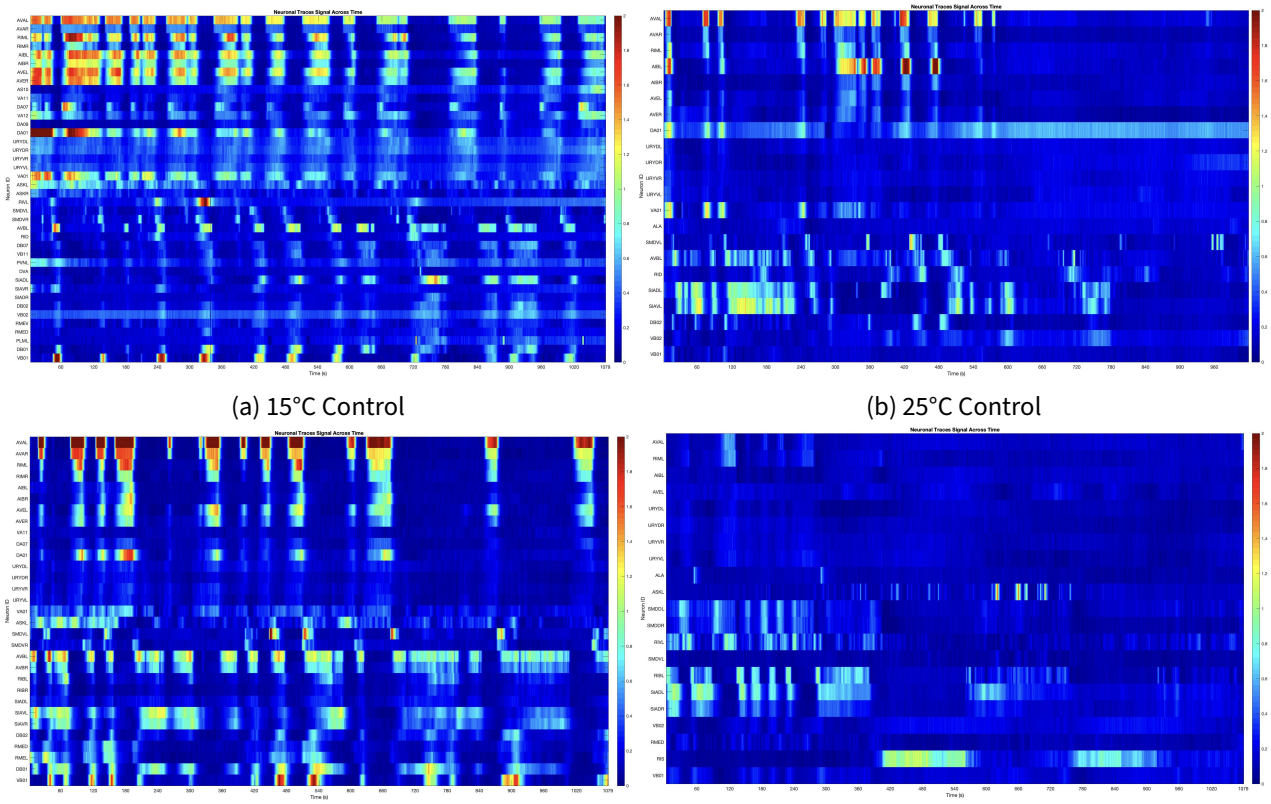
Figure 24: Performance Parameter Comparison. *a-d.Performance parameters, rise/fall time, peak time, overshoot, and settling time are compared between conditions (heating and cooling) (n=6) and between categories (n=3) within conditions,i.e., for heating the 5°C step size from 20°C to 25°C is compared with 10°C step size from 15°C to 25°C; for cooling, the 5°C step size is compared with the 10°C step size from 25°C to 15°C. Plots show mean ± Std. *p<0.05, **p<0.01, ***p<0.001, ***p<0.0001, ns p>0.05.*

		Cooling vs Heating	Heating (5°C vs 10°C step size)	Cooling (5° vs 10°C step size)
Settling time	p-value	0.94	0.20	1.00
	statistic	19.0	8.0	4.0
Rise/fall time	p-value	0.70	0.40	0.70
	statistic	15.0	7.0	3.0
Peak time	p-value	0.70	0.20	0.70
	statistic	21.0	8.0	3.0
Overshoot	p-value	0.13	0.70	0.08
	statistic	8.0	6.0	9.0

Table 4: Performance Parameter Comparison. *Mann Whitney U test results for the performance parameters comparisons, rise/fall time, peak time, overshoot, and settling time between conditions (heating and cooling) and between categories within conditions,i.e., for heating the 5°C step size from 20°C to 25°C is compared with 10°C step size from 15°C to 25°C; for cooling, the 5°C step size is compared with the 10°C step size from 25°C to 15°C.*

3.3 Whole-brain Imaging Data

Whole-brain recordings were performed for TCS validation. In total 4 datasets of each condition, 15°C control, 25°C control, 15°C mutant, 25°C mutant were obtained. Figure 25 shows the heat maps of neuronal activity as variation in signal intensity ($\Delta F/F_0$) for each neuron over time. Representative example recordings for each condition are shown.



(a) 15°C Control

(b) 25°C Control

(c) 15°C *cha-1* mutant

(d) 25°C *cha-1* mutant

Figure 25: Heat maps of neuronal activity traces. *a-d. Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min. 15°C condition on the left (a, c) and 25°C condition on the right (b, d).*

Heat maps of the control line at 15°C shows firing patterns and synchronous activity in clusters related to reverse and forward movements, aligning with findings by Uzel et al. (2022)[57] and Kato et al. (2015)[3](Figure 25a). At a higher temperature of 25°C, the control line shows a slight reduction in activity but maintains robust neuronal activity and synchronisation with key neurons (Figure 25b). The mutant line at 15°C shows similar activity pattern to the control line at the same temperature, indicating that the mutation does not significantly affect neuronal activity under permissive conditions (Figure 25c). However, at 25°C, the mutant line shows reduced neuronal activity compared to both the control line at 25°C and the mutant line at 15°C. Neuronal activity is sparse or missing (Figure 25d). The Root Mean Square (RMS) quantification further supports these observations (Figure 26). In both mutant conditions (15°C and 25°C) and the control line at 25°C, fewer neurons were identified compared to the control line at 15°C (Figure 25).

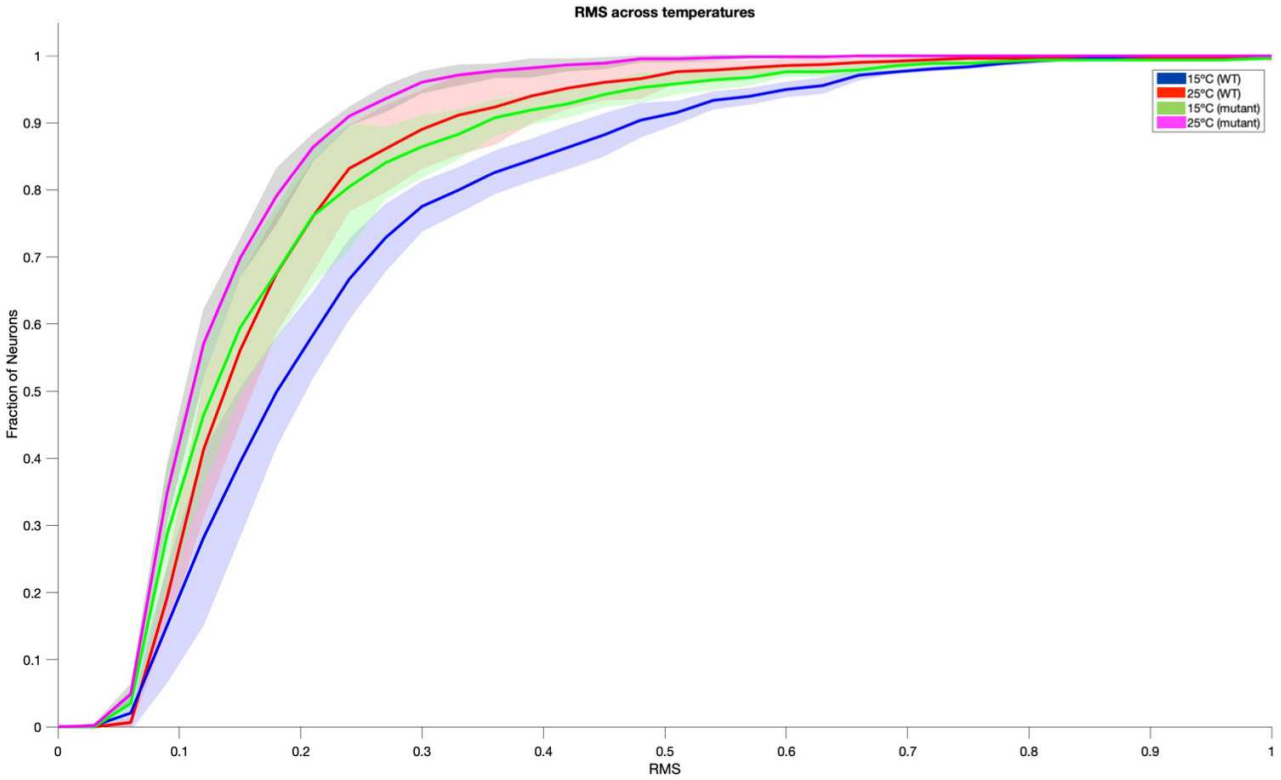


Figure 26: Root Mean Square (RMS) across conditions. *Cumulative RMS values of neuronal activity for identifiable neurons of each condition (n=4).*

RMS values for the control line at 15°C and 25°C show similar curve shapes, suggesting that temperature does not drastically alter the overall RMS distribution of neuronal activity. However, a leftward shift in the curve at 25°C indicates a greater fraction of neurons with lower RMS values indicating decreased activity levels at the higher temperature. The mutant at 25°C shows lower RMS values for the majority of neurons, suggesting that the mutation at this restrictive temperature leads to decreased overall neuronal activity and potentially impaired function. At 15°C, the mutant neurons are more active than at 25°C but still less active overall compared to control line at 15°C.

4 Discussion

The aim of this thesis was to design, develop and evaluate a control system to regulate the temperature of *C. elegans* ts mutants, enabling validity and reliability of experimental outcomes. Step response tests showed that the system reacts adequately to changes in set point, the system is adaptive in cooling and heating conditions for different step sizes and showed high accuracy in maintaining target temperatures. To achieve optimal performance for a given control task, there is often a trade-off of one characteristic for others to meet their requirements. This tuning process involved balancing each parameter (proportional, integral and derivative) to get the optimal system response for the TCS to achieve both quick responsiveness and steady-state accuracy. However, reaching optimal performance, also depends on the system being controlled. As in the case of the TCS, the PDMS' thermal properties predetermine the shape of the response curve. It indicates a certain level of system thermal inertia as it only slowly adapts to changes in control input. Thermal inertia combines both thermal conductivity (or resistance) and thermal capacity. It refers to how quickly a material or system can change temperature in response to an energy input or removal. A system with a higher thermal inertia will have a slower response to changes in the control input, leading to longer rise, longer peak times and consequently longer settling times[19, p.52-57],[58]. This means that in scenarios where the system exhibits low thermal conductivity, (the thermal conductivity of PDMS is in the order of 0.16 W/mK (inherently resistant to heat flow)) [38],[39],[40], the speed of response is inherently slower due to the slower propagation of heat. Due to these properties overshoot is expected as the plant cannot adapt quickly enough to changes in output signals. However, settling is reached relatively quickly after, which indicates that the system is fast in correcting any deviation from the set point. After reaching settling, the temperature remains stable within the tolerance range of 1°C ($\pm$ 0.5°C form the set point). The TCS is designed to obtain temperature readings directly from the experimental site (worm-channel). This direct approach is particularly important when considering the thermal properties of PDMS. There exist significant temperature differences between the surface of the PDMS block and the internal channels. Indirect or external temperature measurements do not accurately reflect the internal environmental conditions. This could lead to potential errors in the experimental findings. Direct temperature measurement ensures that the conditions experienced by the worm are consistent with the experimental requirements. The system shows good control performance. It is, however, not possible to generalize the observed results, due to the small sample sizes used (total: n=12; category: n=3 and condition: n=6). This increases the risk of failing to detect any true effect. A larger sample size is necessary to draw more robust conclusions. The trends indicated, nevertheless offer insights into system dynamics under various situations and conditions and suggest potential areas of optimization.

System performance: cooling vs. heating
The system generally shows similar levels of performance in both heating and cooling modes. It tends to be more precise and quicker when cooling compared to heating. The tendency to perform worse when heating is potentially due to the thermal properties of the system and materials involved but also possibly due the system design (to remove heat). During cooling mode, active heatsinks help to dissipate heat from the non-controlling side of the Peltier elements (working principle of Peltier cells, see 2). Active cooling can contribute to a more efficient heat dissipation which leads to the observed quicker fall and peak times and better mitigation of overshoot. This makes the system more predictive and controllable during cooling modes. In contrast, with the active heatsinks shut down during heating, the temperature only passively dissipates from the system without any active system to counteract and makes the system more variable.

System performance: different step sizes (5°C and 10°C)
The controller adapts to heating and cooling demands, this means, the larger the step size, the stronger the control effort (the larger is the proportional corrective action). This is shown by the prolonged maximum control output when the error is larger (Figure 22). In theory, for linear systems, performance measures like rise/fall time, overshoot, and settling time are independent of the step size. However, in practical applications, especially with systems like the TCS with thermal dynamics and active components like Peltier

elements, these performance metrics can depend on the input step size[20, p.151]. This non-linearity might arise from varying operational efficiency of the Peltier elements at different temperatures[43] or thermal properties of the PDMS. In the TCS when the step size is smaller, it shows that the system tends to be more stable whereas, if the step size is larger, the system has slightly longer adjustment times and slightly higher overshoot. However, this is more pronounced in heating conditions, likely due to the active heat dissipation during cooling modes as discussed. In addition, due to the increased control action needed to overcome the larger temperature difference (10°C) larger overshoot is more likely to occur[20, p.294].

Post-settling dynamics - maintaining set temperature
After the system has reached the set point, the control output shows minor fluctuations. These fluctuations represent the ongoing adjustments by the PI controller to maintain the temperature at the set point. The consistent control effort observed after the transient phase, implies that the system has reached steady state, maintaining the temperature within the set point range. Both, cooling and heating conditions are precise in maintaining the target temperature post-settling. Further, the system shows a high level of stability irrespective of step amplitude. Fluctuations in the temperature data suggest occasional instability or sensor measurement noise(Figure 22). However, these minor fluctuations post-settling don't affect the overall system performance.

TCS validation & whole brain data
To validate the TCS, whole-brain imaging data of *C. elegans* under different temperature conditions were analyzed. At 15°C, heat maps of control worms show normal network function with synchronised activity in reverse and forward neurons, consistent with previously reported data by Uzel et. al. (2020)[57] and Kato et. al. (2015)[3]. At 25°C, control worms exhibit similar activity with slight changes suggesting the neural network remains robust to temperature changes within the range of 15°C to 25°C (the viable or growth-temperature in labs of *C. elegans* ranges from 12°C to 25°C[2]). However, higher temperatures may slightly alter activity due to increased metabolic rates or stress responses[2],[59]. For *cha-1* mutant worms, the neuronal activity at 15°C resembles that of the control line at 15°C, suggesting preserved network function and normal ACh-production at this permissive temperature. However, the mutation could still introduce slight variations in neural dynamics[6]. At 25°C *cha-1* mutant worms show severe disruptions with drastically reduced neuronal activity, indicating impaired ACh production and instability in neural interaction and function [4],[2],[5]. The RMS analysis supports these findings, with a left shift for the control line at 25°C, suggesting reduced neuronal activity compared to 15°C due to higher the temperature even in the absence of the mutation[2],[59]. Similarly, in the mutant line at 25°C shows lower RMS values for the majority of neurons, indicating that the restrictive temperature significantly reduces neuronal activity and network function. These observations indicate both, the temperature sensitivity of neuronal activity and the impact of the *cha-1* mutation, particularly the impaired ACh production at the higher temperature[6]. At 15°C in the mutant line, neurons are more active compared to mutant at 25°C, which aligns with expectations, as at mutant at the permissive temperature has a phenotype similar to the control line where ACh-production is closer to wild-type[6]. However, even at this permissive temperature, there is lower overall activity compared to control line at the same temperature. This suggests that the mutation by itself might have some minor effects on neuronal activity patterns.

The trends seen in the existing data provide initial insights into how temperature and mutation may influence neuronal activity. However, these findings are preliminary and further recordings are needed to draw more reliable conclusions.

4.1 Limitations and Outlook

TCS thermal inertia
The TCS implemented was primarily evaluated based on its ability to reach and maintain set point temperatures over time. This means, stability and precision were prioritized over speed. The focus is more on maintaining specific conditions not on how quickly those conditions are reached. However, for experiments

where the focus is on rapid temperature changes, this particular system may not be well suited due to the inherent system inertia.

Problems with increasing response speed - make system faster
Proportional control action in a the TCS determines how the system reacts to differences between the current temperature and the set point. When a significant change is made, such as adjusting the set point from 15°C to 25°C, the proportional control is what causes the control system to apply a strong initial corrective effort. This is because the proportional part of the controller multiplies the difference, or error, between the set point and the current temperature by the proportional gain to determine the output[18, p.3-4]. If the system's output is capped at a maximum level, in this case 1500, the proportional control will push the output to this maximum value when a large error is detected. The output signal will not exceed this cap, but it will remain at the maximum level until the error decreases to a point where the proportional gain times the error equals less than the maximum output. Increasing the proportional gain to increase the system's heating or cooling rates would lead to several problems. Increasing the proportional gain in a situation where the maximum output is already being reached would not increase the actual output any further due to the cap. However, it would change the system's behavior in response to smaller errors. Specifically, for any given error, a higher proportional gain would result in a higher output. This would make the system react more aggressively to even small deviations from the set point[20, p.294]. This aggressive response can potentially lead to instability, such as pronounced oscillations, because the system will be quick to max out its output with even slight temperature differences. This is why it's crucial to balance the proportional gain with the integral action. A faster response could lead to instability, so there's often a trade-off to balance response speed and potential instability to reach optimal performance within the limits of the system's maximum output capacities and its thermal properties[23, p.70, 85],[19, p.141].

Temperature range
The performance of the TCS was assessed within a specific temperature range (10°C range - 15°C to 25°C). It is not known how the system might behave or how accurate it would be outside of the evaluated range. The operational efficiency and accuracy of components like thermistor and Peltier elements can vary across different temperatures[43],[42]. This means, optimal system operation and performance is restricted to a specific temperature range. The TCS might therefore, not be applicable for experimental conditions with more extreme temperatures.

4.2 Conclusion

Results from the step response tests indicate that the system responds adequately to set point changes and maintains target temperatures with a high level of accuracy. It shows a good balance between responsiveness and steady-state accuracy, which suggests a well-tuned PI-controller. The performance remains consistent across different experimental conditions such as varying step sizes and modes (cooling and heating). Key performance parameters like rise time, peak time, overshoot and settling time can vary slightly depending on the specific condition. However, these variations do not affect overall accuracy, which indicates that the system can adapt to and performs well under varying experimental conditions. Further, the neural activity patterns observed align with expected neural responses at specific temperatures. This indicates that the TCS's temperature readings match the temperature experienced by the worm.

5 References

References

1. White, J. G., Southgate, E., Thomson, J. N. & Brenner, S. The structure of the nervous system of the nematode Caenorhabditis elegans: the mind of a worm. *Phil. Trans. R. Soc. Lond* **314,** 340 (1986).
2. Corsi, A. K., Wightman, B. & Chalfie, M. A Transparent window into biology: A primer on *Caenorhabditis elegans*, (June 18, 2015). *WormBook, ed. The C. elegans Research Community, WormBook.* `http://www.wormbook.org` (2015).
3. Kato, S. *et al.* Global brain dynamics embed the motor command sequence of Caenorhabditis elegans. *Cell* **163,** 656–669 (2015).
4. Pereira, L. *et al.* A cellular and regulatory map of the cholinergic nervous system of C. elegans. *Elife* **4,** e12432 (2015).
5. Duerr, J. S., Han, H.-P., Fields, S. D. & Rand, J. B. Identification of major classes of cholinergic neurons in the nematode Caenorhabditis elegans. *Journal of Comparative Neurology* **506,** 398–408 (2008).
6. Duerr, J. S., McManus, J. R., Crowell, J. A. & Rand, J. B. Analysis of Caenorhabditis elegans acetylcholine synthesis mutants reveals a temperature-sensitive requirement for cholinergic neuromuscular function. *Genetics* **218,** iyab078 (2021).
7. Arthur, L. L. *et al.* Rapid generation of hypomorphic mutations. *Nature communications* **8,** 14112 (2017).
8. Schrödel, T., Prevedel, R., Aumayr, K., Zimmer, M. & Vaziri, A. Brain-wide 3D imaging of neuronal activity in Caenorhabditis elegans with sculpted light. *Nature methods* **10,** 1013–1020 (2013).
9. Kaplan, H. S., Thula, O. S., Khoss, N. & Zimmer, M. Nested neuronal dynamics orchestrate a behavioral hierarchy across timescales. *Neuron* **105,** 562–576 (2020).
10. Zimmer, M. *et al.* Neurons detect increases and decreases in oxygen levels using distinct guanylate cyclases. *Neuron* **61,** 865–879 (2009).
11. Mavromatidis, L. E., Bykalyuk, A., El Mankibi, M., Michel, P. & Santamouris, M. Numerical estimation of air gaps' influence on the insulating performance of multilayer thermal insulation. *Building and Environment* **49,** 227–237 (2012).
12. Juanicó, L. E. & González, A. D. Thermal insulators with multiple air gaps: Performance, cost and embodied impacts. *Journal of Building Engineering* **12,** 188–195 (2017).
13. Casquillas, G. V. *et al.* Fast microfluidic temperature control for high resolution live cell imaging. *Lab on a Chip* **11,** 484–489 (2011).
14. Cornaglia, M. *et al.* Automated longitudinal monitoring of in vivo protein aggregation in neurodegenerative disease C. elegans models. *Molecular neurodegeneration* **11,** 1–13 (2016).
15. Miralles, V., Huerre, A., Malloggi, F. & Jullien, M.-C. A review of heating and temperature control in microfluidic systems: techniques and applications. *Diagnostics* **3,** 33–67 (2013).
16. Dos-Reis-Delgado, A. A. *et al.* Recent advances and challenges in temperature monitoring and control in microfluidic devices. *Electrophoresis* **44,** 268–297 (2023).
17. Shafiei, N. *et al.* Development of portable air conditioning system using peltier and seebeck effect. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* **8,** 97–100 (2016).
18. Visioli, A. *Practical PID control* (Springer Science & Business Media, 2006).
19. Bolton, W. *Control systems* (Newnes, 2002).
20. Åström, K. J. & Murray, R. M. *Feedback systems: an introduction for scientists and engineers* (Princeton university press, 2021).
21. Åström, K. J. & Hägglund, T. *PID Controller* (Instrument Society of America, 1995).

22. Sung, S. W., Lee, J. & Lee, I.-B. *Process identification and PID control* (John Wiley & Sons, 2009).
23. Dogan, I. *PID-based Practical Digital Control* (Elektor International Media B.V., 2022).
24. Lee, C.-C. Fuzzy logic in control systems: fuzzy logic controller. I. *IEEE Transactions on systems, man, and cybernetics* **20,** 404–418 (1990).
25. Larsen, P. M. Industrial applications of fuzzy logic control. *International Journal of Man-Machine Studies* **12,** 3–10 (1980).
26. Pappis, C. P. & Mamdani, E. H. A fuzzy logic controller for a trafc junction. *IEEE Transactions on Systems, Man, and Cybernetics* **7,** 707–717 (1977).
27. Castro, J. L. Fuzzy logic controllers are universal approximators. *IEEE transactions on systems, man, and cybernetics* **25,** 629–635 (1995).
28. Krstic, M. & Smyshlyaev, A. Adaptive control of PDEs. *Annual Reviews in Control* **32,** 149–160 (2008).
29. Qin, S. J. & Badgwell, T. A. *An overview of industrial model predictive control technology* in *AIche symposium series* **93** (1997), 232–256.
30. Schwenzer, M., Ay, M., Bergs, T. & Abel, D. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology* **117,** 1327–1349 (2021).
31. Holkar, K. & Waghmare, L. M. An overview of model predictive control. *International Journal of control and automation* **3,** 47–63 (2010).
32. Psaltis, D., Sideris, A. & Yamamura, A. A. A multilayered neural network controller. *IEEE control systems magazine* **8,** 17–21 (1988).
33. Lin, S. & Goldenberg, A. A. Neural-network control of mobile manipulators. *IEEE Transactions on Neural networks* **12,** 1121–1133 (2001).
34. Braganza, D., Dawson, D. M., Walker, I. D. & Nath, N. A neural network controller for continuum robots. *IEEE transactions on robotics* **23,** 1270–1277 (2007).
35. Wang, L. *PID Control System Design and Automatic Tuning using MATLAB: Simulink* (John Wiley & Sons Ltd).
36. Patel, V. V. Ziegler-nichols tuning method: Understanding the pid controller. *Resonance* **25,** 1385–1397 (2020).
37. Ziegler, J. G. & Nichols, N. B. Optimum settings for automatic controllers. *Transactions of the American society of mechanical engineers* **64,** 759–765 (1942).
38. Wei, J. *et al.* Enhanced thermal conductivity of polydimethylsiloxane composites with carbon fiber. *Composites Communications* **17,** 141–146 (2020).
39. Vlassov, S. *et al.* Thermal, mechanical, and acoustic properties of polydimethylsiloxane filled with hollow glass microspheres. *Materials* **15,** 1652 (2022).
40. Xu, K. *et al. Design and simulation of MEMS thermal and vibration isolator based on PDMS beam arrays* in *2016 IEEE 11th Annual International Conference on Nano/Micro Engineered and Molecular Systems (NEMS)* (2016), 213–216.
41. *fritzing electronics made easy* https://fritzing.org. (accessed: 09.04.2024).
42. *Mouser Electronics* https://www.mouser.at/ProductDetail/Littelfuse/USP12837?qs=JeIcUl65ClCk%252BLylkTUBmA%3D%3Df. (accessed: 10.04.2024).
43. *Laird thermal systems* https://lairdthermal.com/products/thermoelectric-cooler-modules/peltier-annular-series/SH10-95-06-L-RT-W12. (accessed: 10.04.2024).
44. Prasad, A. *et al.* Applications of light-emitting diodes (LEDs) in food processing and water treatment. *Food Engineering Reviews* **12,** 268–289 (2020).
45. *Aakash + BYJU's* https://byjus.com/jee/semiconductors/. (accessed: 09.04.2024).

46. Unruh, R., Schafmeister, F. & Böcker, J. *11kW, 70kHz LLC Converter Design with Adaptive Input Voltage for 98% efficiency in an MMC* in *2020 IEEE 21st Workshop on Control and Modeling for Power Electronics (COMPEL)* (2020), 1–8.

47. *Sparkfun* https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf. (accessed: 09.04.2024).

48. Nedelkovski, D. *How to Mechatronics* https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/?utm_content=cmp-true. (accessed: 09.04.2024).

49. *PJRC* https://www.pjrc.com/store/teensy41.html. (accessed: 10.04.2024).

50. *Arduino* https://www.arduino.cc. (accessed: 10.04.2024).

51. *Thermistor* https://en.wikipedia.org/wiki/Thermistor#Steinhart%E2%80%93Hart_equation. accessed: 23.09.2024.

52. Beauregard, B. *Arduino PID Library* https://github.com/br3ttb/Arduino-PID-Library. Version 1.2.1. 2021.

53. *LAUDA* https://www.lauda.de/en/. (accessed: 09.04.2024).

54. Kintel, M. *OpenSCAD The Programmers Solid 3D CAD Modeller* https://openscad.org. (accessed: 26.07.2024).

55. Dana, H. *et al.* High-performance calcium sensors for imaging activity in neuronal populations and microcompartments. *Nature methods* **16,** 649–657 (2019).

56. Pokala, N., Liu, Q., Gordus, A. & Bargmann, C. I. Inducible and titratable silencing of Caenorhabditis elegans neurons in vivo with histamine-gated chloride channels. *Proceedings of the National Academy of Sciences* **111,** 2770–2775 (2014).

57. Uzel, K., Kato, S. & Zimmer, M. A set of hub neurons and non-local connectivity features support global brain dynamics in C. elegans. *Current Biology* **32,** 3443–3459 (2022).

58. Karlsson, J., Wadsö, L. & Öberg, M. A conceptual model that simulates the influence of thermal inertia in building structures. *Energy and Buildings* **60,** 146–151 (2013).

59. Lee, S.-J. & Kenyon, C. Regulation of the longevity response to temperature by thermosensory neurons in Caenorhabditis elegans. *Current biology* **19,** 715–722 (2009).

6 Supplementary material

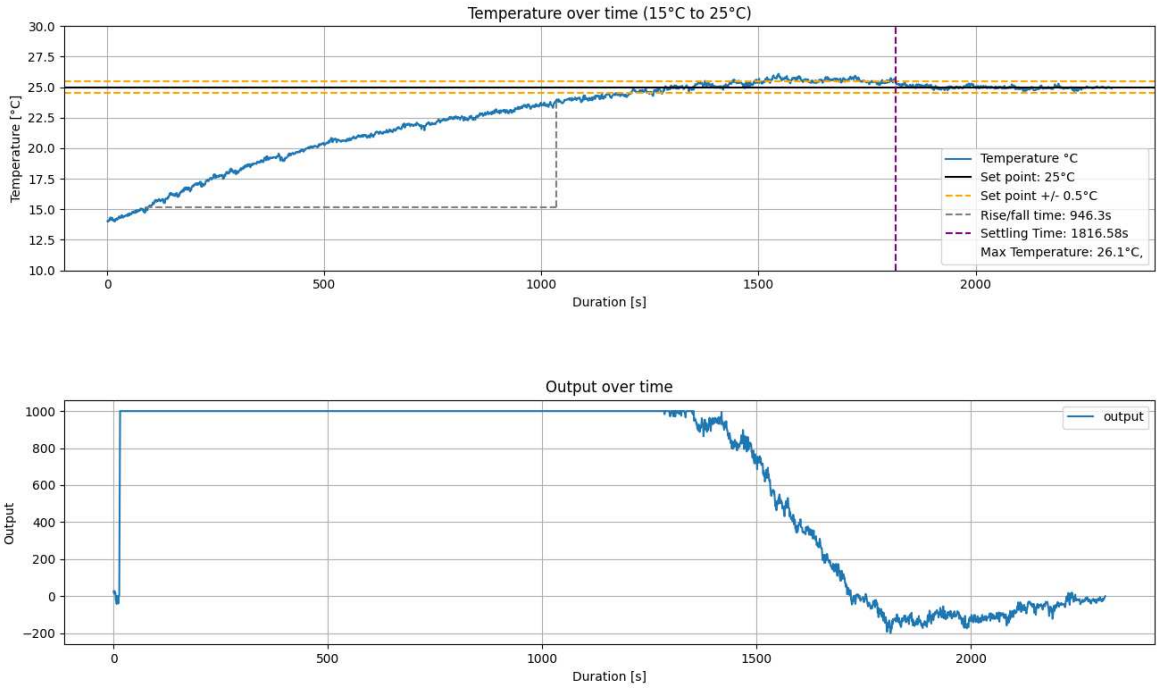


Figure 27: Set point Step Response. *Response dynamics of the TCS for heating, step size of 10°C from 15°C to 25°C.*

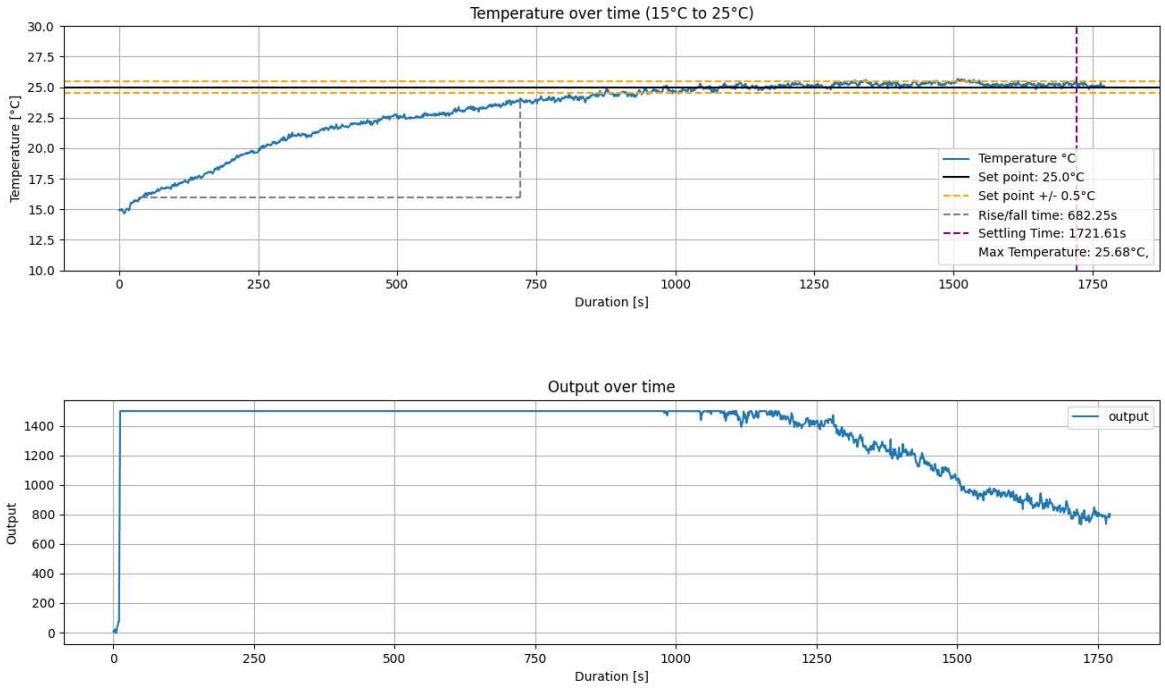


Figure 28: Set point Step Response. *Response dynamics of the TCS for heating, step size of 10°C from 15°C to 25°C*

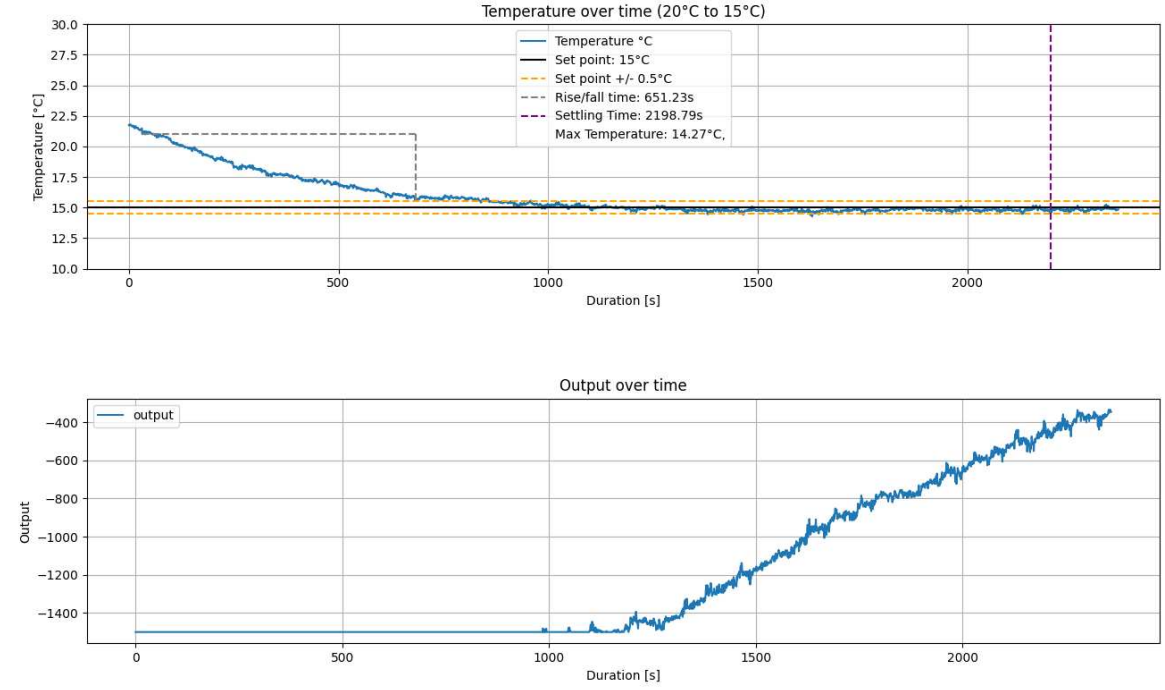


Figure 29: Set point Step Response. *Response dynamics of the TCS for cooling, step size of 5°C from 20°C to 15°C*

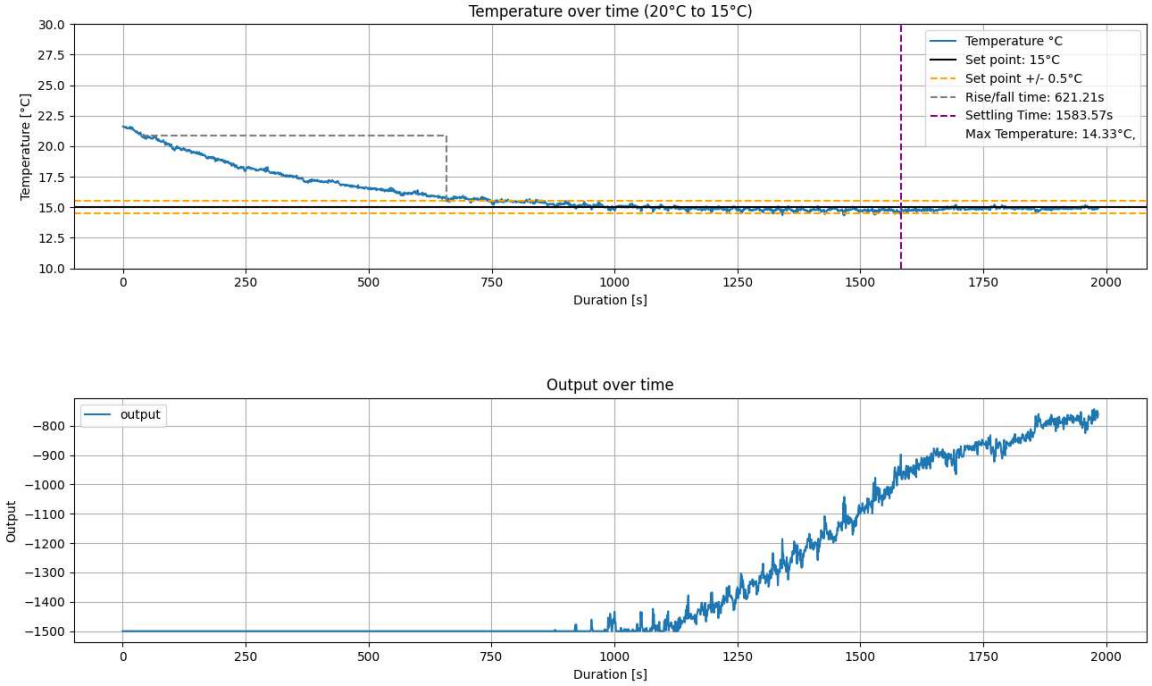


Figure 30: Set point Step Response. *Response dynamics of the TCS for cooling, step size of 5°C from 20°C to 15°C*

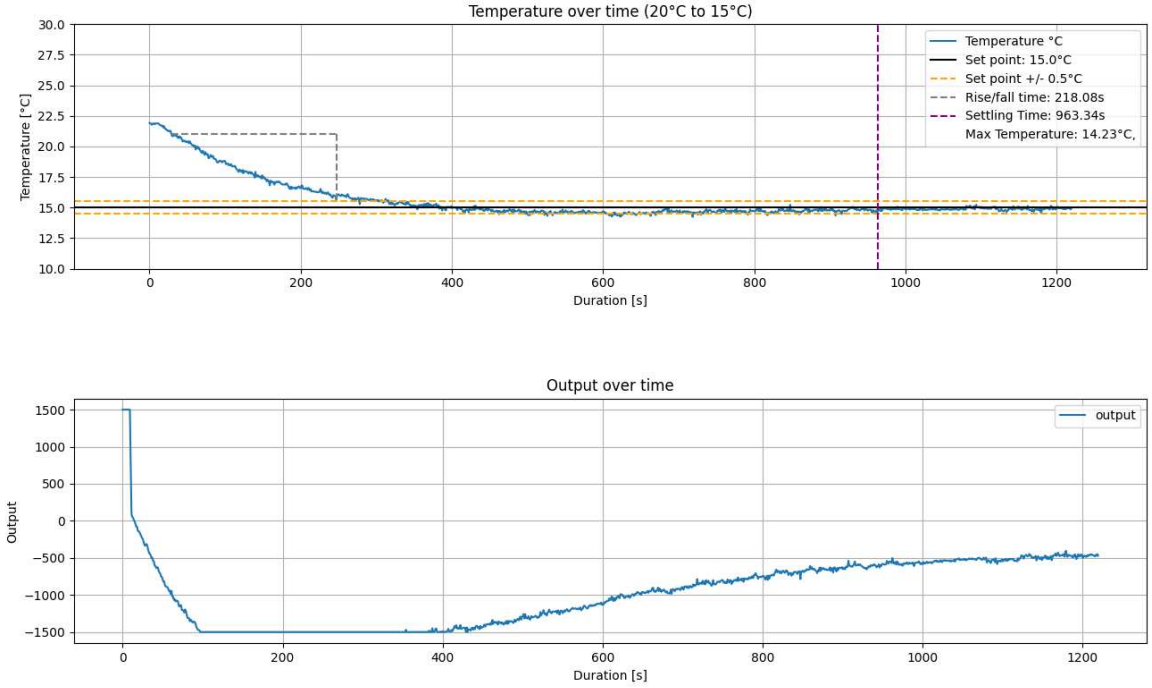


Figure 31: Set point Step Response. *Response dynamics of the TCS for cooling, step size of 5°C from 20°C to 15°C*

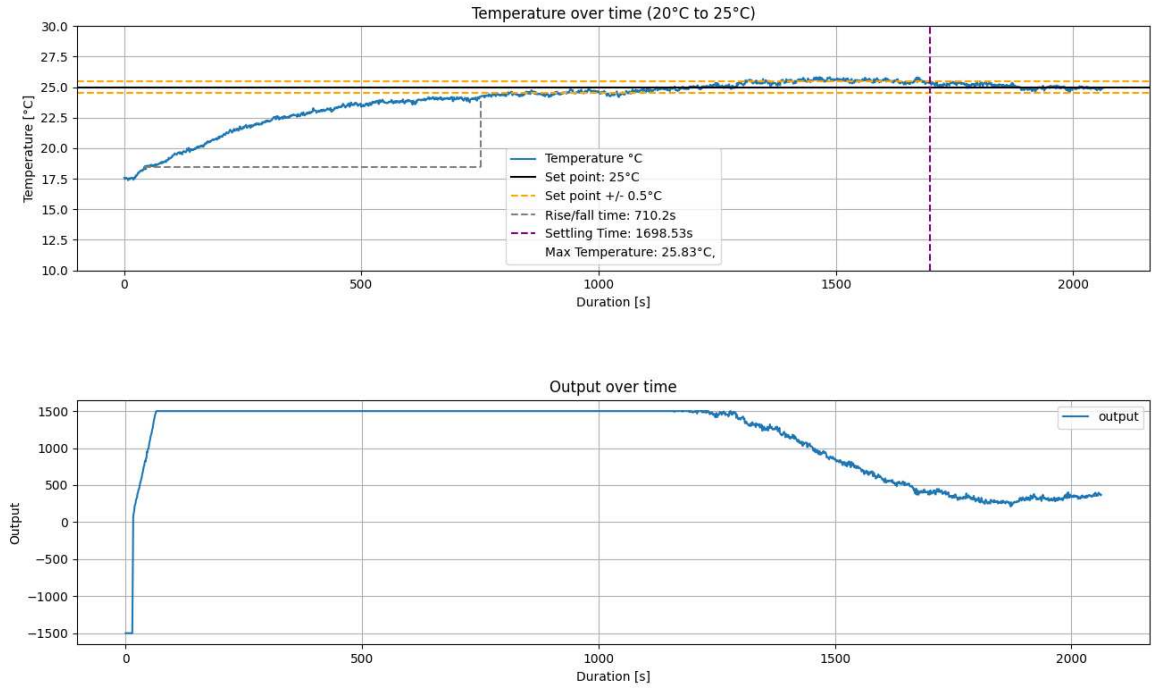


Figure 32: Set point Step Response. *Response dynamics of the TCS for heating, step size of 5°C from 20°C to 25°C*

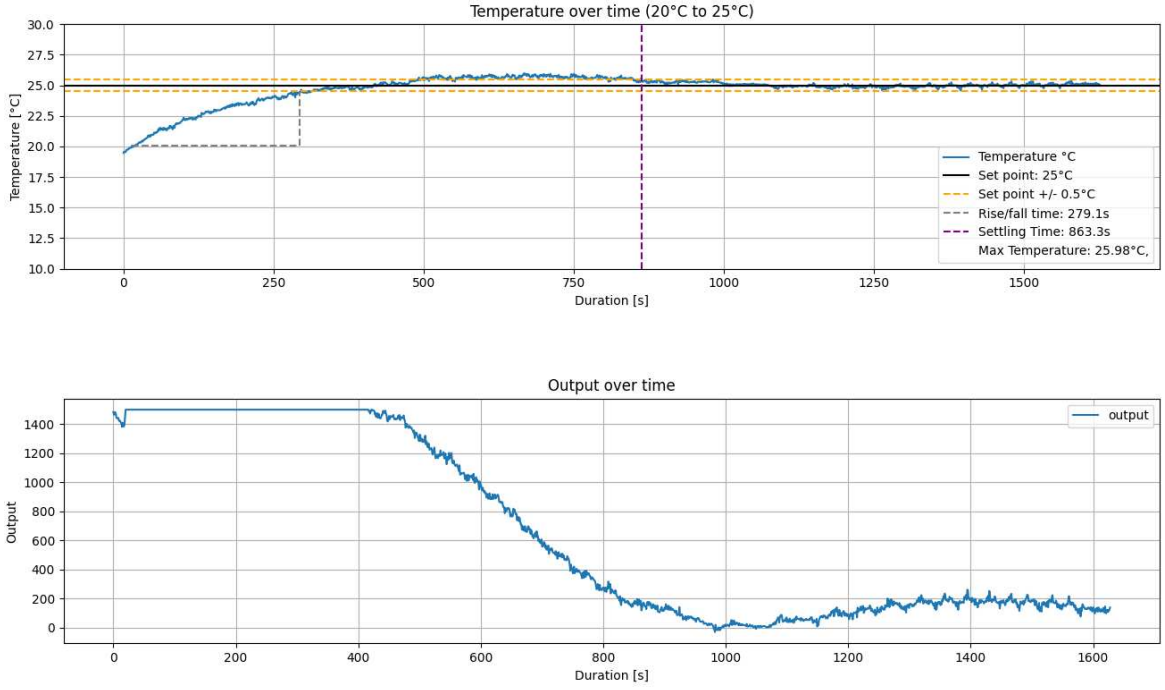


Figure 33: Set point Step Response. *Response dynamics of the TCS for heating, step size of 5°C from 20°C to 25°C*

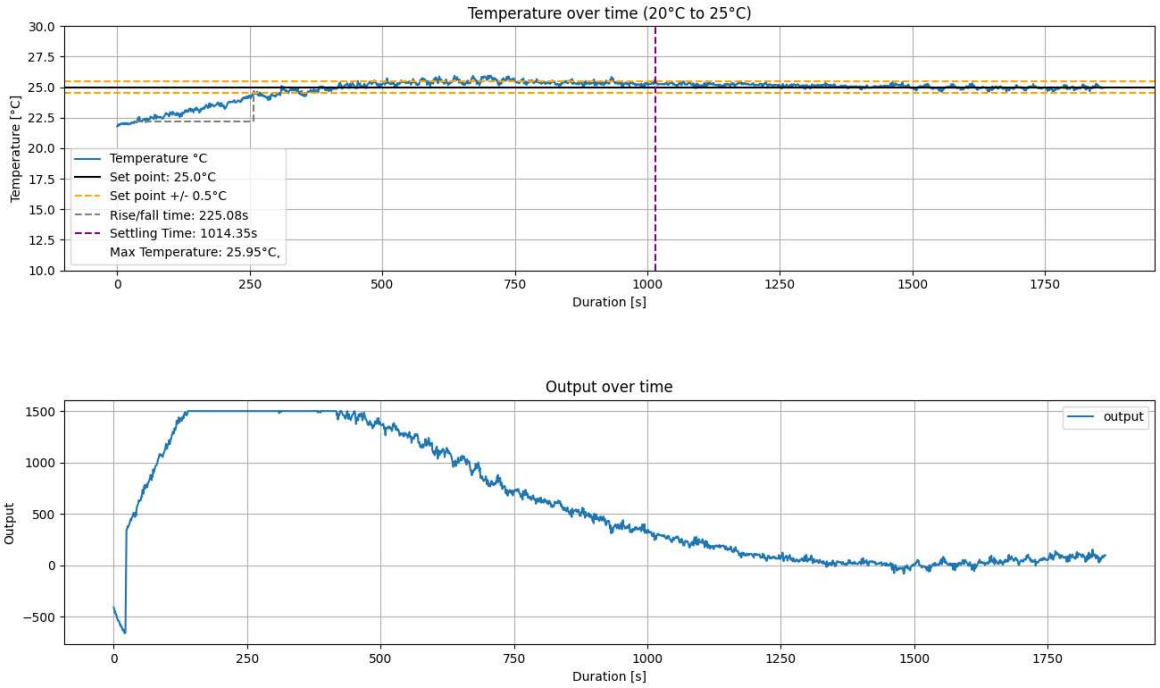


Figure 34: Set point Step Response. *Response dynamics of the TCS for heating, step size of 5°C from 20°C to 25°C*

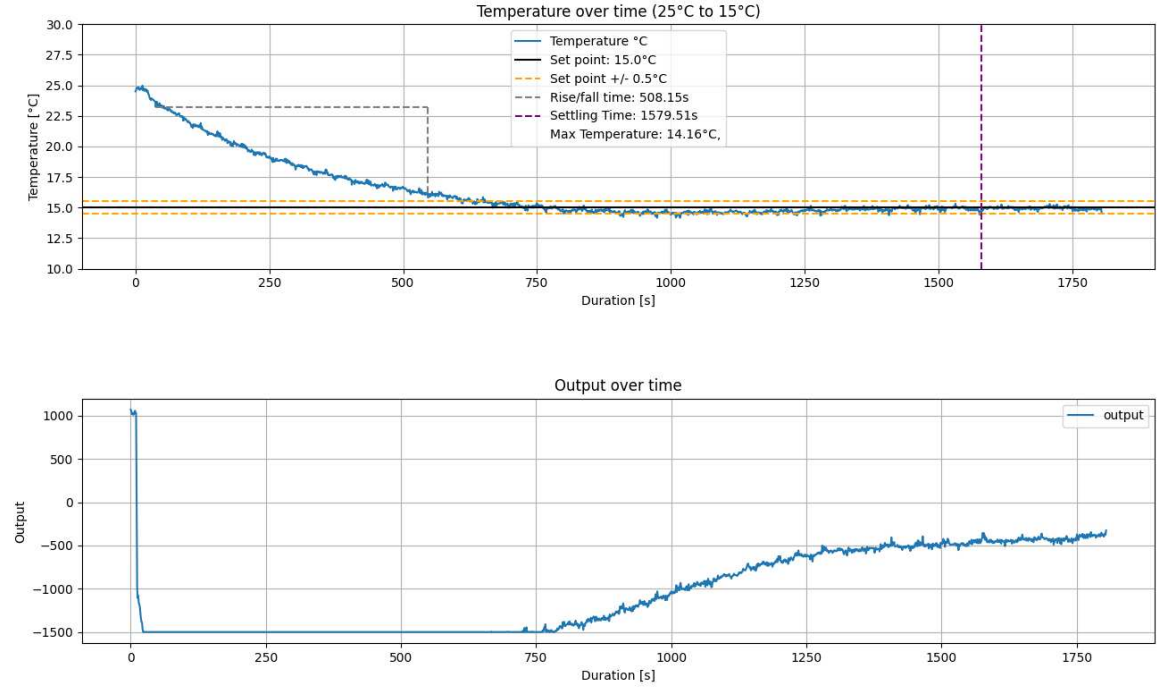


Figure 35: Set point Step Response. *Response dynamics of the TCS for cooling, step size of 10°C from 25°C to 15°C*

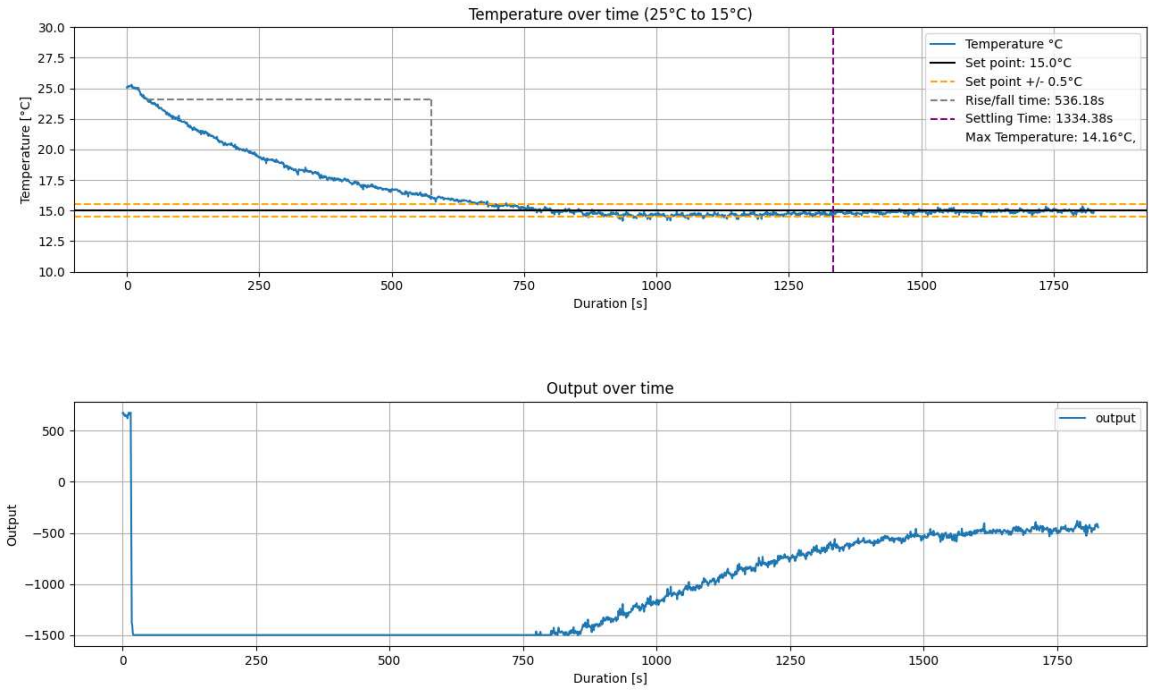


Figure 36: Set point Step Response. *Response dynamics of the TCS for cooling, step size of 10°C from 25°C to 15°C*

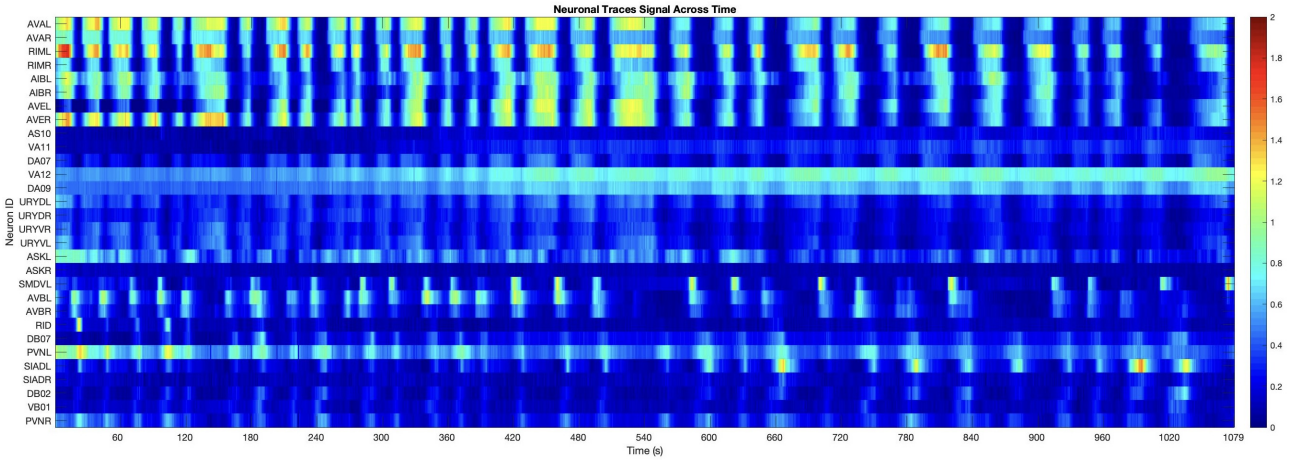


Figure 37: Heat map 15°C control. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

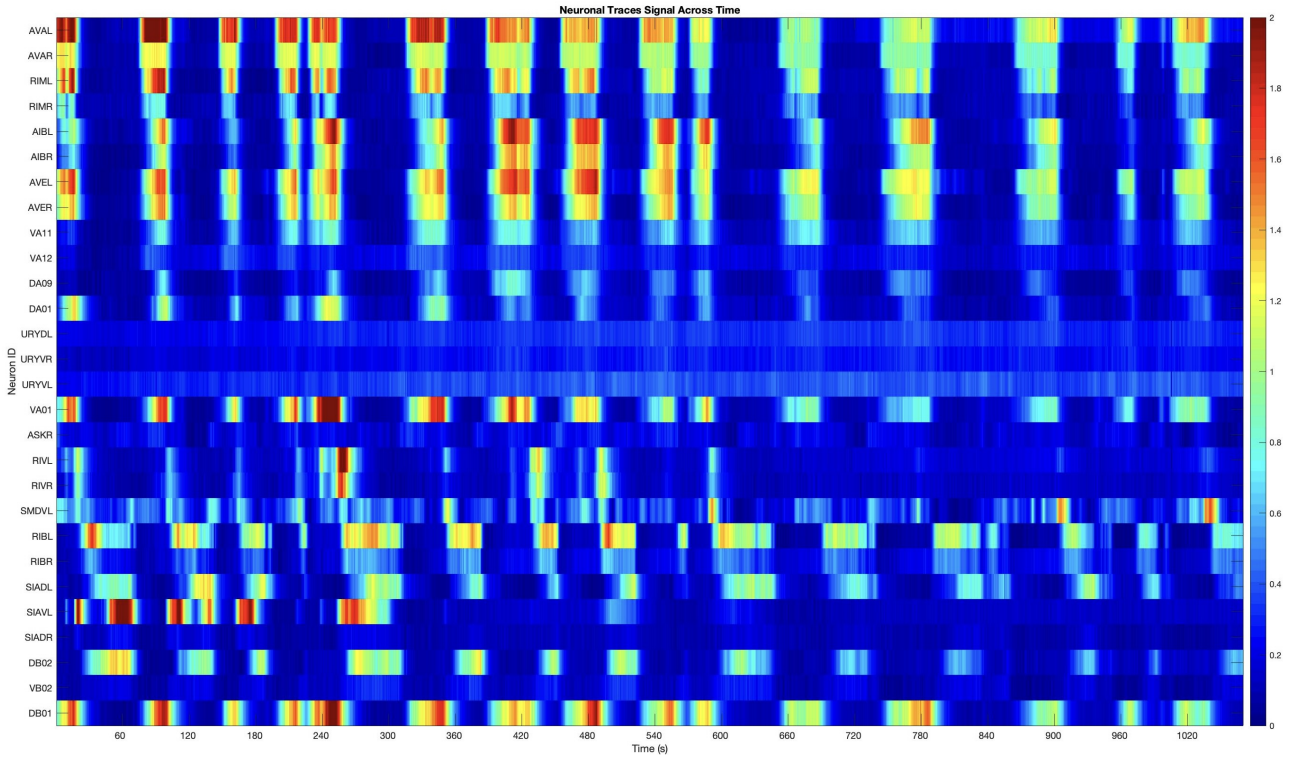


Figure 38: Heat map 15°C control. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

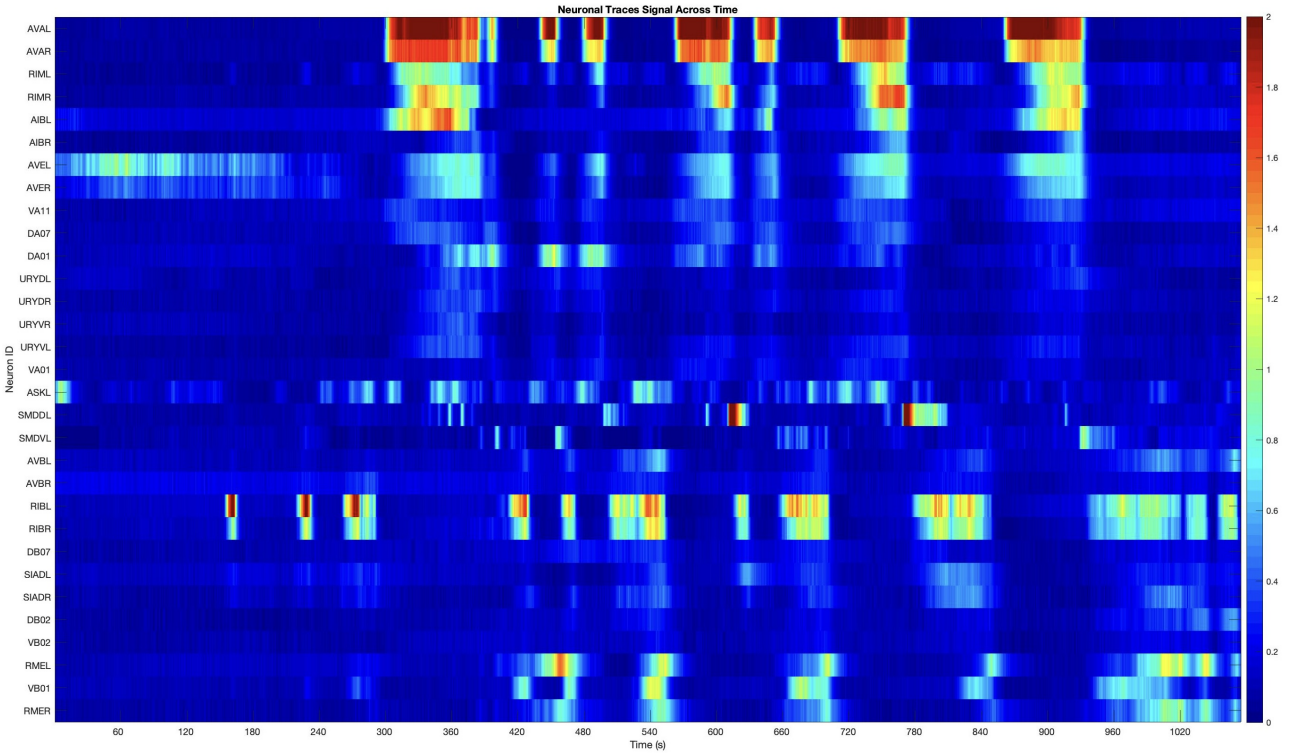


Figure 39: Heat map 15°C mutant. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

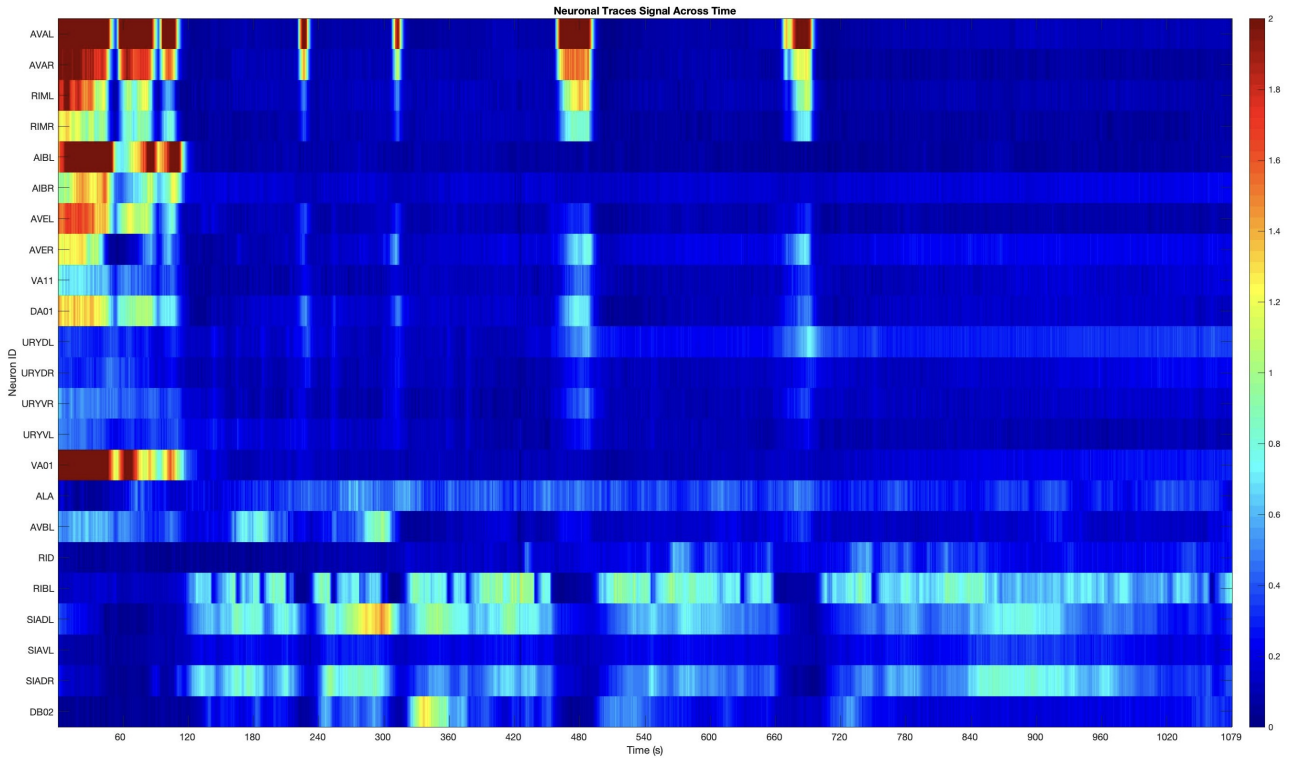


Figure 40: Heat map 15°C mutant. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

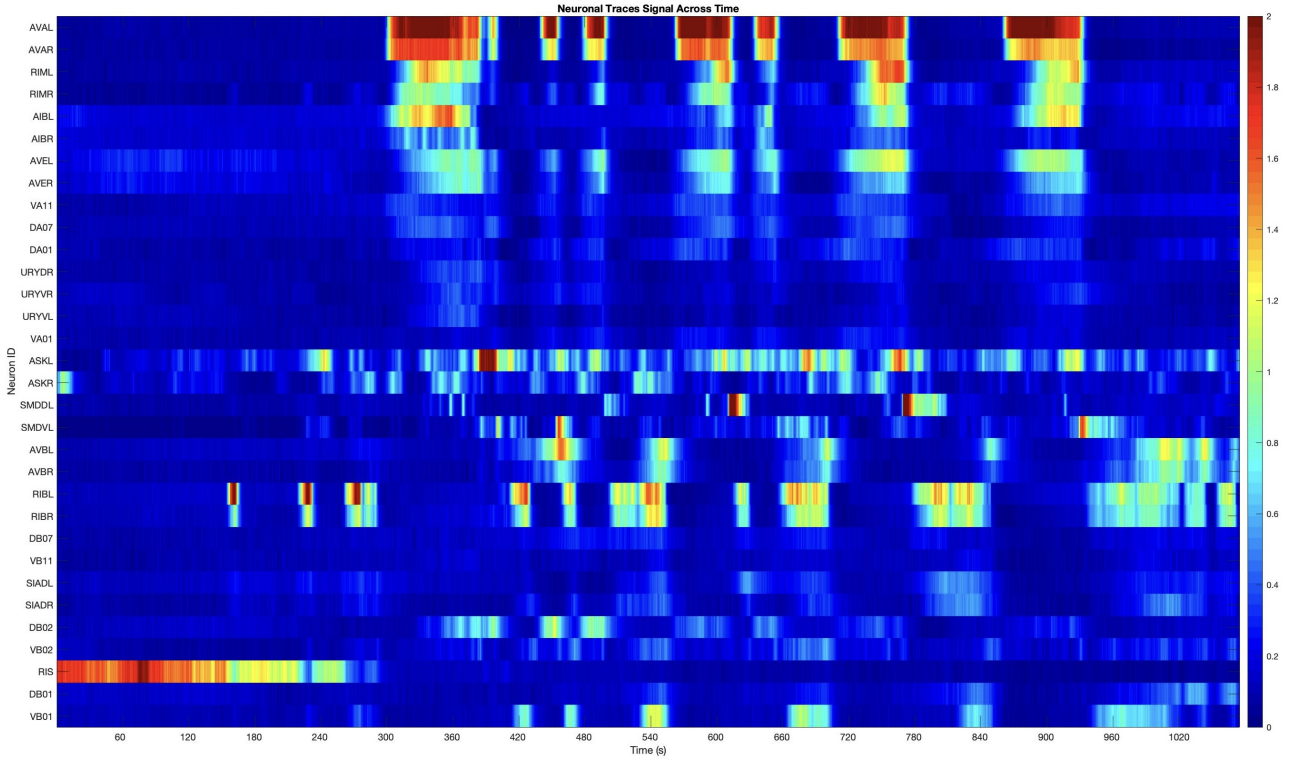


Figure 41: Heat map 15°C mutant. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

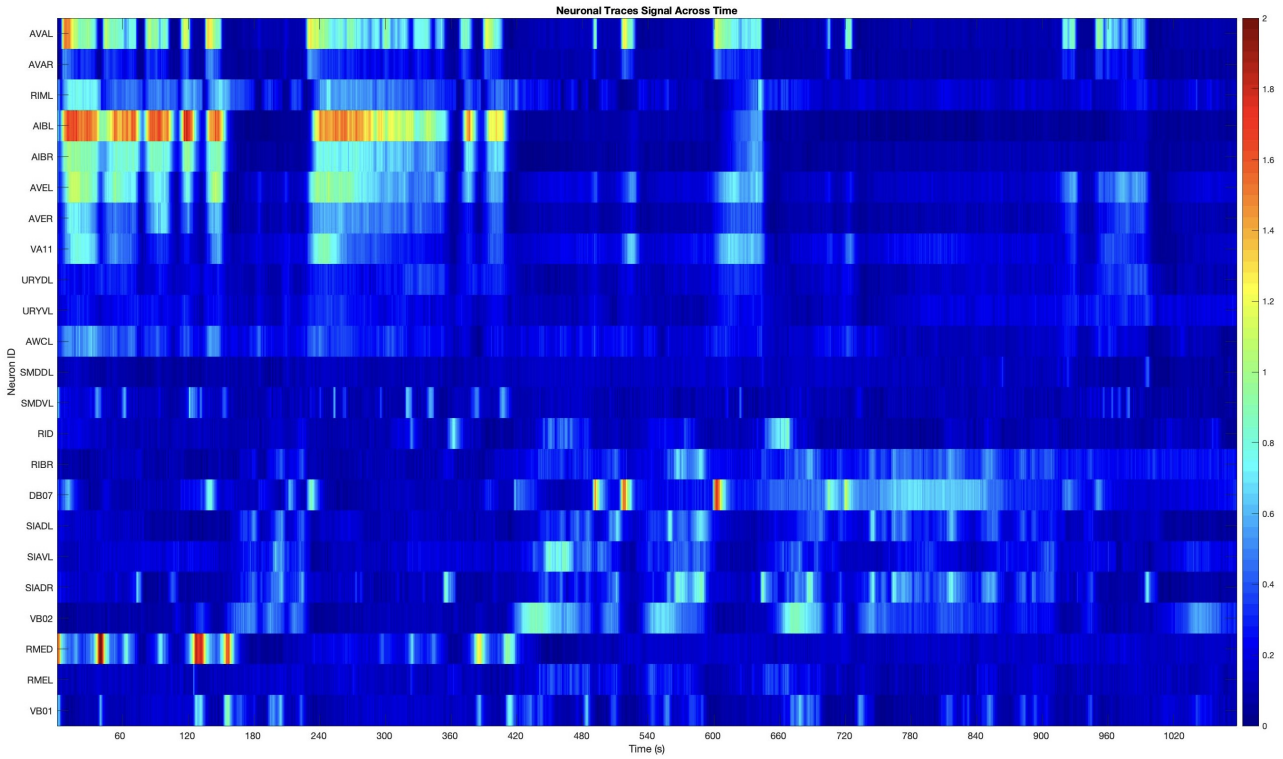


Figure 42: Heat map 25°C control. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

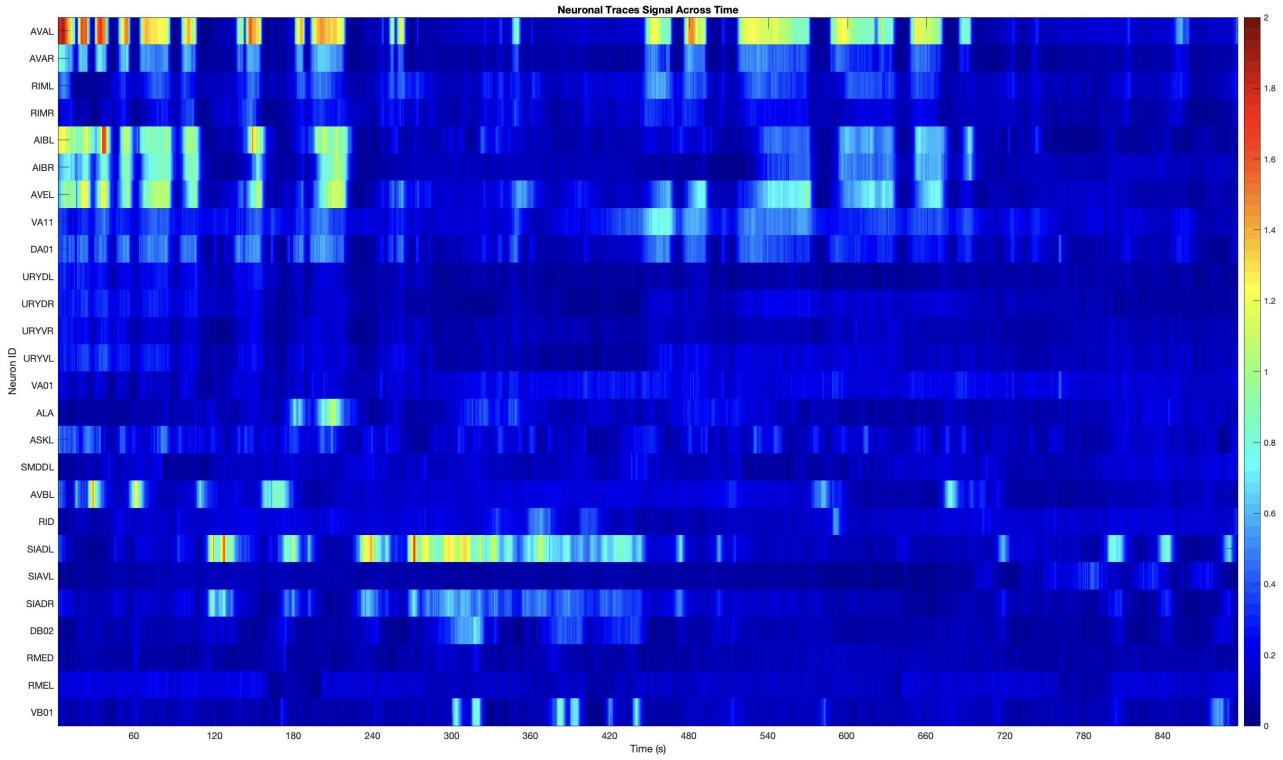


Figure 43: Heat map 25°C control. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

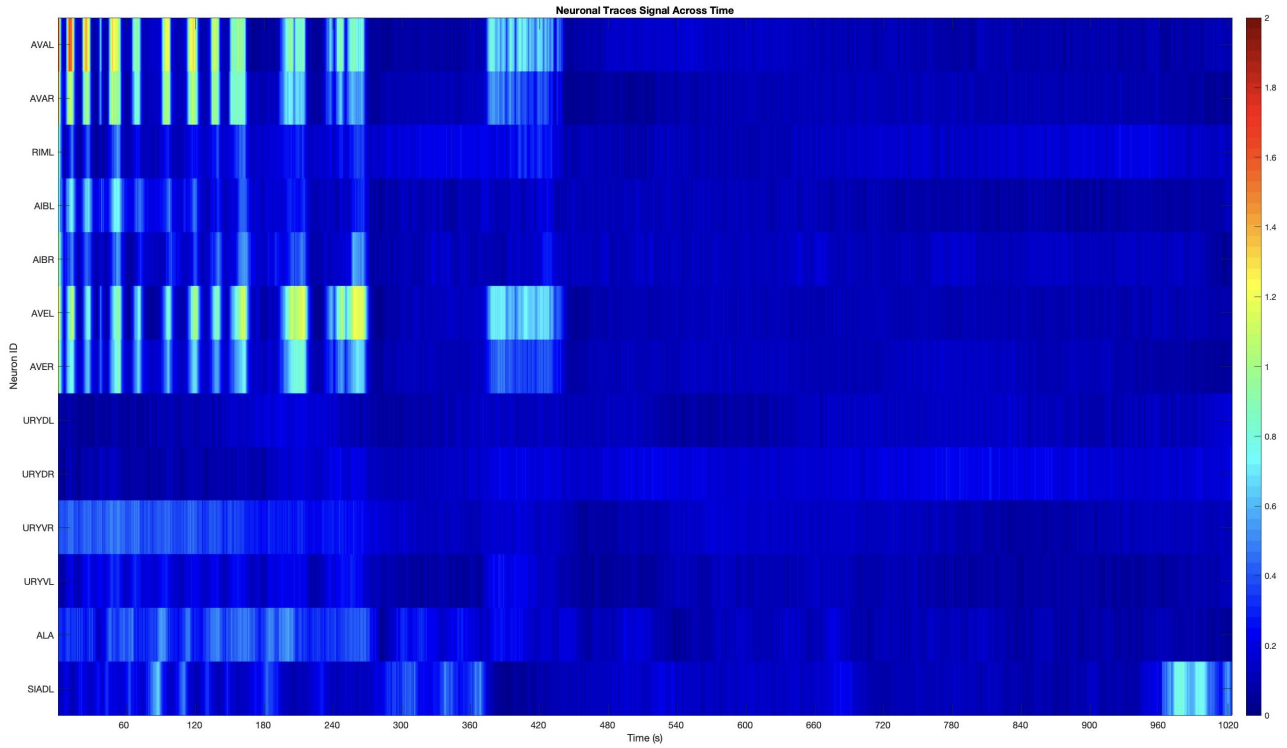


Figure 44: Heat map 25°C control. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

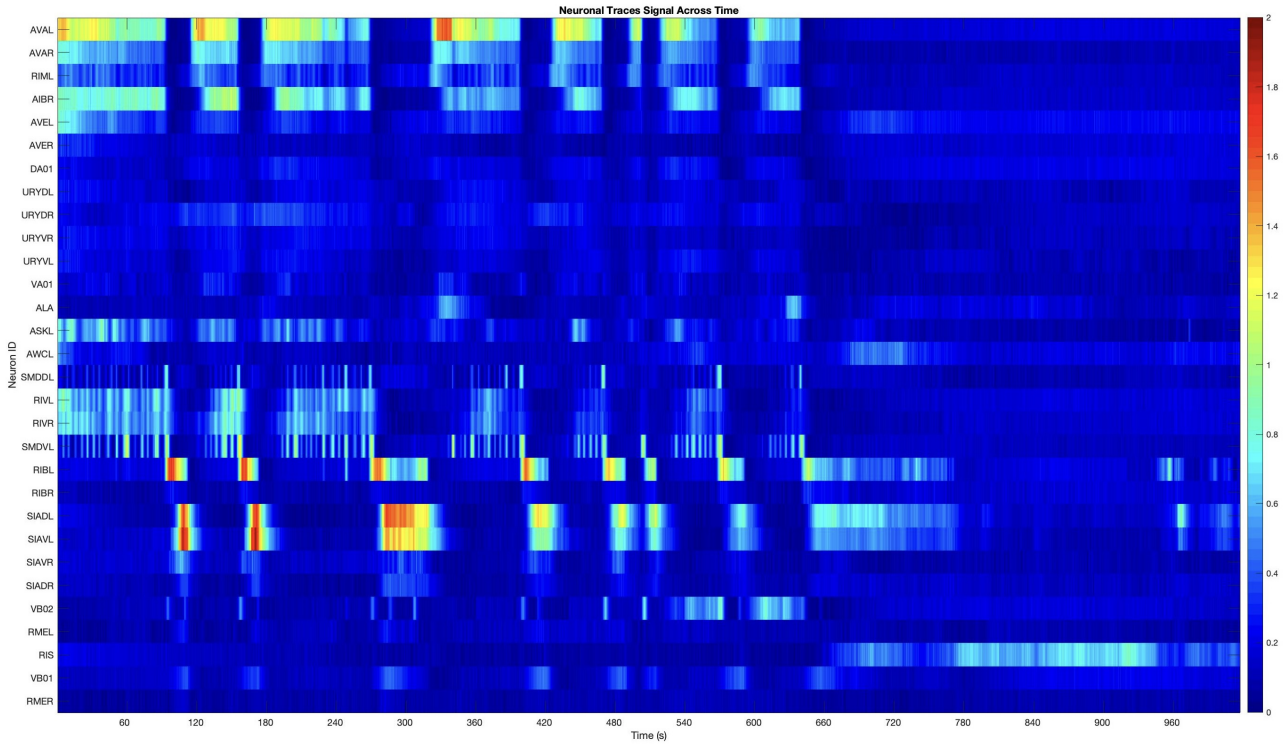


Figure 45: Heat map 25°C mutant. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

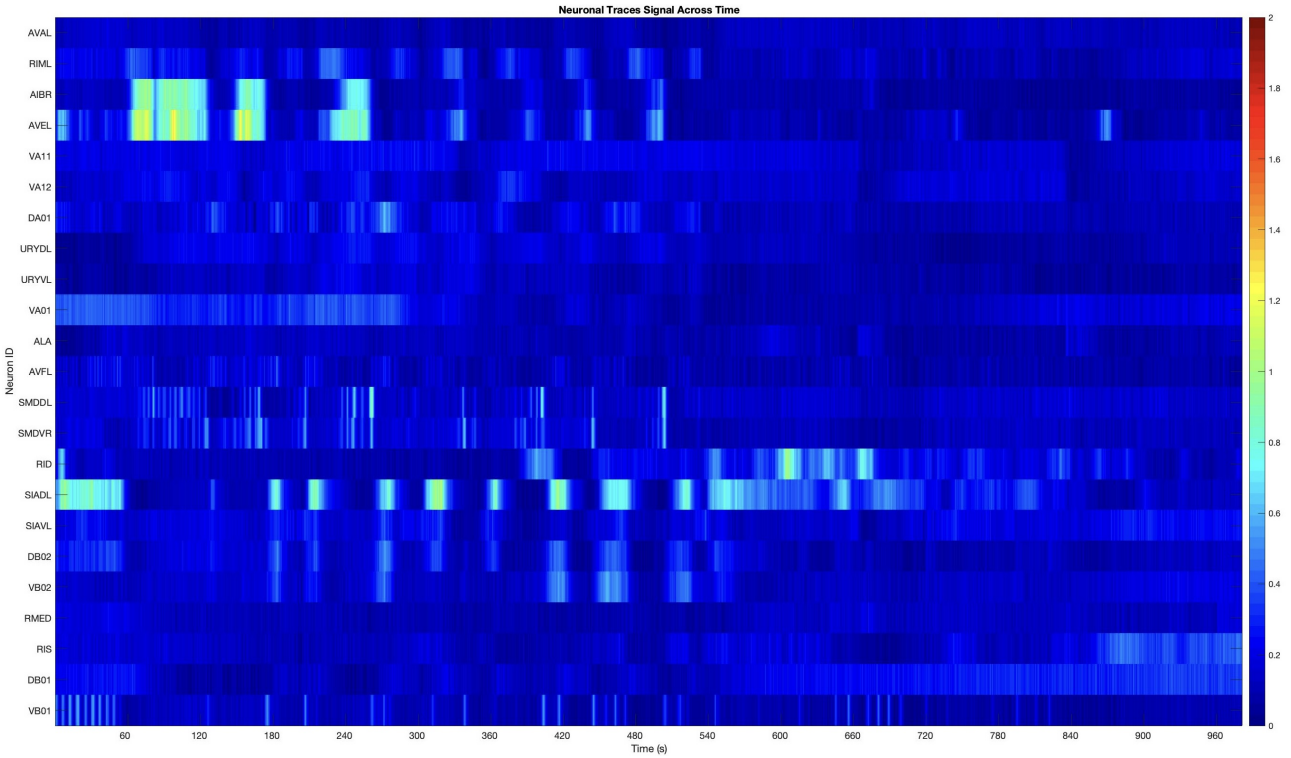


Figure 46: Heat map 25°C mutant. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

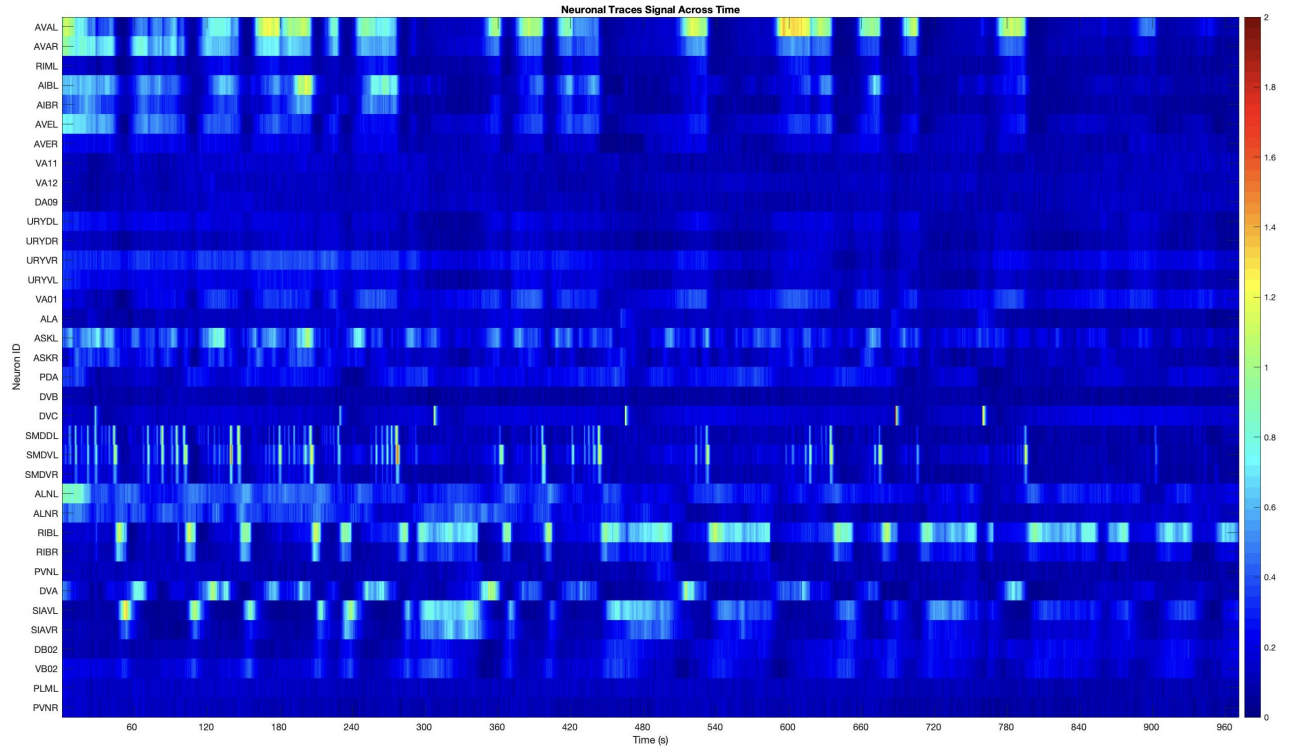


Figure 47: Heat map 25°C mutant. *Heat plots of fluorescence ($\Delta F/F$) time series of identified neurons, one neuron per row. Recording-time 18min.*

	Run	Start temperature [°C]	Temperature mean [°C]	Temperature std	Setpoint [°C]	Inverted	Rise/fall time [s]	Peak time [s]	Settling time [s]	Peak temperature [°C]	Peak Temperature Overshoot [°C]
15°C-25°C	Run 571	14.96	23.16	2.72	25.00	FALSE	682.25	1,507.53	1,721.61	25.68	0.68
	Run 578	15.18	23.74	2.46	25.00	FALSE	505.16	934.29	1,070.33	26.00	1.00
	Run 518	14.02	22.71	3.20	25.00	FALSE	946.30	1,546.43	1,816.58	26.10	1.10
20°C-25°C	Run 529	19.47	24.70	1.21	25.00	FALSE	279.10	667.23	863.30	25.98	0.98
	Run 569	21.79	24.84	0.83	25.00	FALSE	225.08	702.24	1,014.35	25.95	0.95
	Run 525	17.57	23.97	1.87	25.00	FALSE	710.20	1,465.45	1,698.53	25.83	0.83
25°C-15°C	Run 570	25.35	16.04	2.41	15.00	TRUE	530.18	1,166.36	1,645.51	14.13	0.87
	Run 575	24.50	16.25	2.47	15.00	TRUE	508.15	974.30	1,579.51	14.16	0.84
	Run 577	25.07	16.36	2.58	15.00	TRUE	536.18	936.31	1,334.38	14.16	0.84
20°C-15°C	Run 572	21.92	15.57	1.66	15.00	TRUE	218.08	718.25	963.34	14.23	0.77
	Run 527	21.62	15.99	1.71	15.00	TRUE	621.21	1,467.53	1,583.57	14.33	0.67
	Run 523	21.76	15.91	1.74	15.00	TRUE	651.23	1,629.58	2,198.79	14.27	0.73

Table 5: TLC Performance. *Inverted: FALSE = Heating, TRUE = Cooling*

7 Appendix

List of Figures

List of Tables