# Temperature Controller Emulator (C#)

## Brief

Implement –

* A software emulator controller component for an ideal temperature controller.
* A user interface to trigger ramping to temperature and display the controller status.
* Three relevant unit tests.

Document –

* A UML of key system abstractions; hint – you may use the Visual Studio built-in tool.

## Temperature Controller – Ramping to Temperature

The temperature controller implements “ramping to temperature” which is comprised of –

* Going from the current temperature TCURRENT [oC] to a user specified target temperature TTARGET [oC] at the specified temperature ramp rate of TRAMP\_RATE [oC/min], and
* Once the target temperature is within the user specified target settling band TTARGET [oC] ± TSETTLING\_BAND [oC], the ramp needs to stabilise (wait at that temperature) for tSETTLING\_TIME [min].

+25

+85

-45

T [oC]

t [min]

*Graph not to scale.*

[⯈BEGIN RAMP]

TCURRENT

TTARGET

TRAMP\_RATE

TSETTLING\_BAND

⭥

tSETTLING\_TIME

Ramp ends; the temperature controller maintains the temperature.

## User Requirements

The users have provided the following indicative UX of what they’d like to see –

Current temperature 25.0 oC  
🡅 Ramping Up / 🡇 Ramping Down / ■ Maintaining

Target temperature \_\_85.0\_\_ oC  
Ramp rate \_\_\_2.0\_\_ oC/min  
Settling band ±\_\_0.25\_ oC  
Settling time \_\_\_5.0\_\_ min  
[⯈ BEGIN RAMP] [◾ CANCEL RAMP]

## Technical Requirements

1. Use Visual Studio, the .NET framework, and C# for all back-end code.
2. Use the provided *Rakon.Test.Core* library –
   1. Define and implement *IAsyncOperationContext* as you see fit.
   2. You may extend the *ITemperatureController* interface if you think that’s helpful.
3. The software emulator controller component must implement the provided *ITemperatureController* interface with a [1 oC/min, 10 oC/min] ramp rate range.
4. The UI can be web or WPF –
   1. Use the MVC pattern for a web UI, or
   2. The MVVC pattern for the WPF UI.
5. The implementation must be thread-safe.
6. Use the provided *SystemTime* functionality instead of the .NET *DateTime*; this allows for emulation of time (speed factor of 60x).
   1. If you feel you need to, extend the *SystemTime* class.
7. The code must be readable (form before function) and must follow the standard Microsoft C# coding standards.

## Bonus – A More Realistic (e.g. Peltier) Temperature Controller

Instead of an ideal temperature controller (straight line ramp and maintaining of temperature), implement a more realistic one (more ‘curvy/fuzzy’ behaviour), such as a Peltier temperature controller.

Suggested parameters –

* Equipment ambient (room) temperature = 25oC
* Typical heat loss factor at ambient temperature = 48 mW/oC
* Typical ramp rate factor at ambient temperature = 11.4 J/oC

**END**