Tango Design Patterns

Tango Workshop @ ICALEPCS 2021

14 October 2021

Andy Götz (ESRF)

Why Patterns ?

TANGO
Connecting things together

“What's new here is that there's nothing new here. Patterns are about what works.”

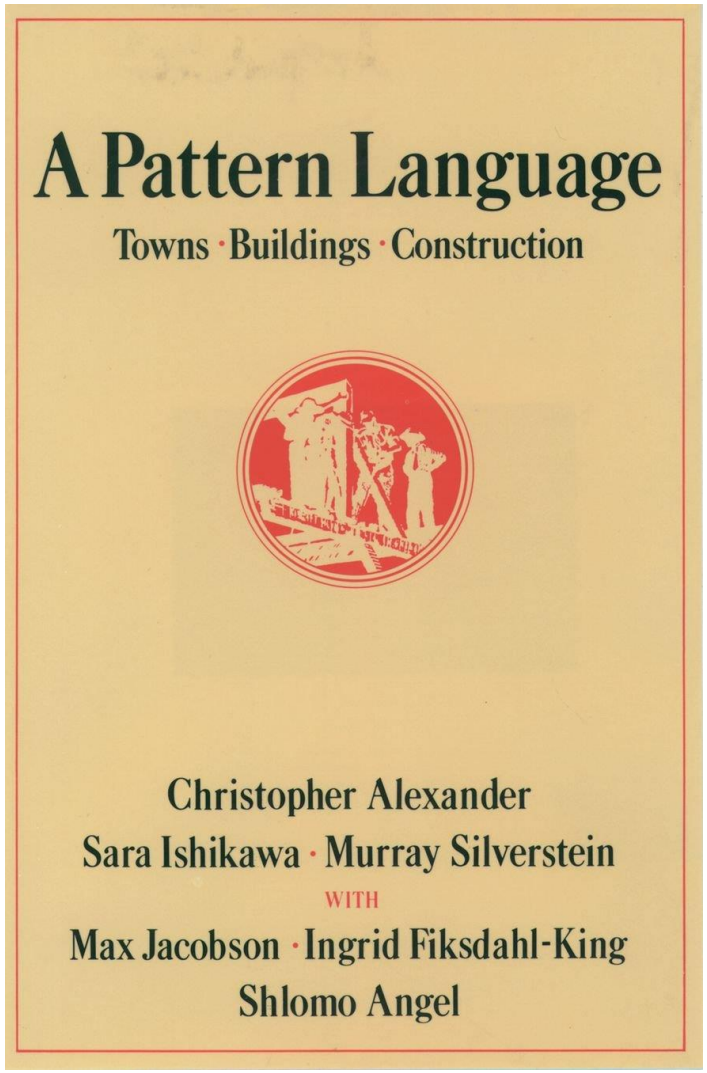
“Patterns give us a way to talk about what works.”

—**Brian Foote**, pattern writer (1997, ix)

A UML Pattern Language by Paul Evitt

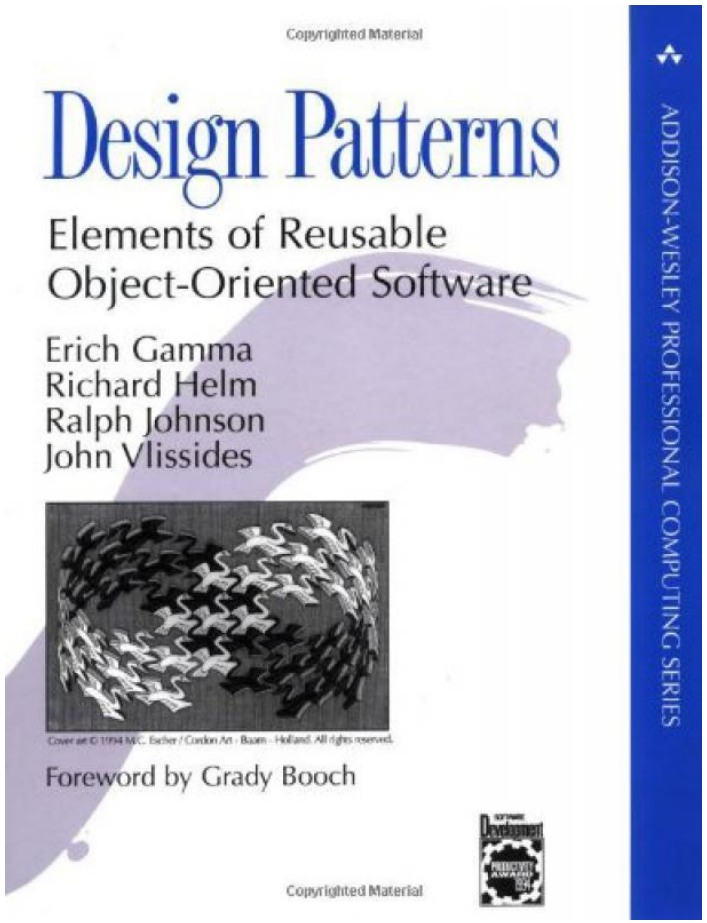
The original Patterns Book

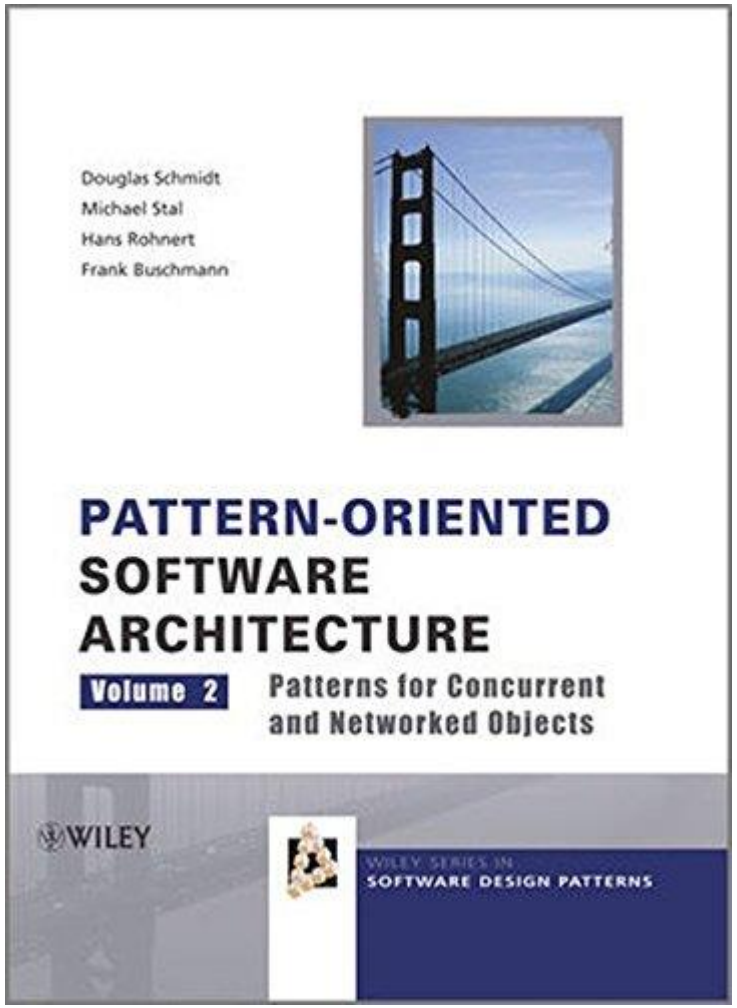
TANGO
Connecting things together

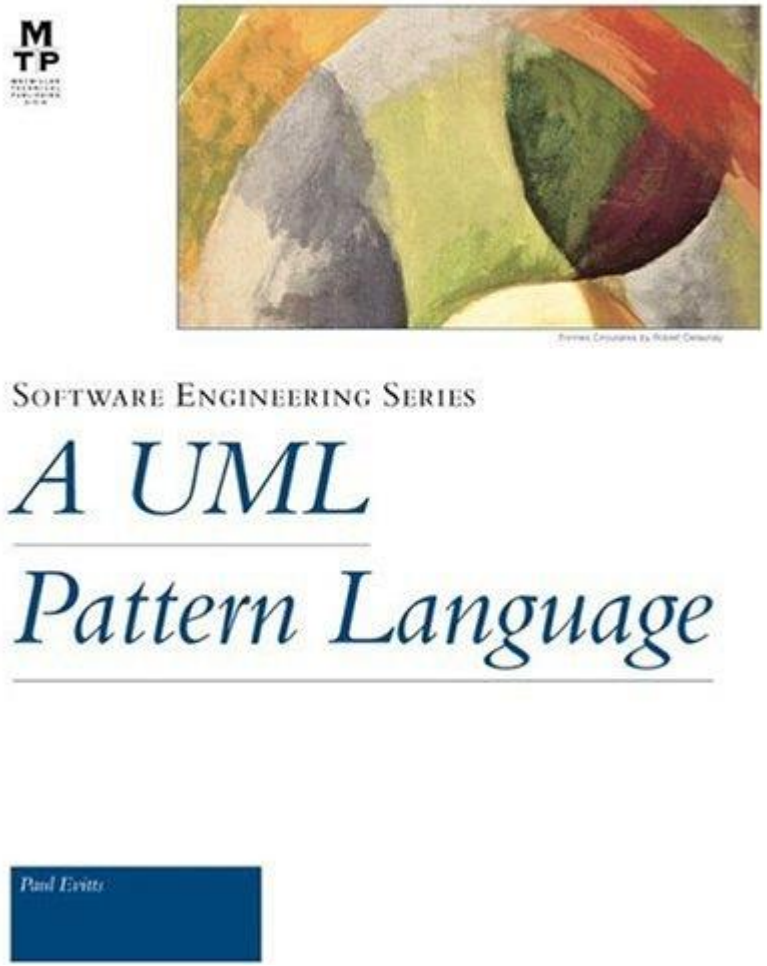


Software Patterns Books

TANGO
Connecting things together







Software Patterns websites

https://www.martinfowler.com/articles/writingPatterns.html

https://wiki.c2.com/?PortlandPatternRepository

http://hillside.net/index.php/a-pattern-language-for-pattern-writing

Tango Patterns Language

TANGO
Connecting things together

1. Context
2. Problem
3. Forces / Constrains
4. Solution
5. Examples

Context

- You are an experienced practitioner in your field. You have noticed that you keep using a certain **solution** to a commonly occurring **problem**. You would like to share your experience with others.

Problem

- How do you share a recurring **solution** to a **problem** with others so that it may be reused?

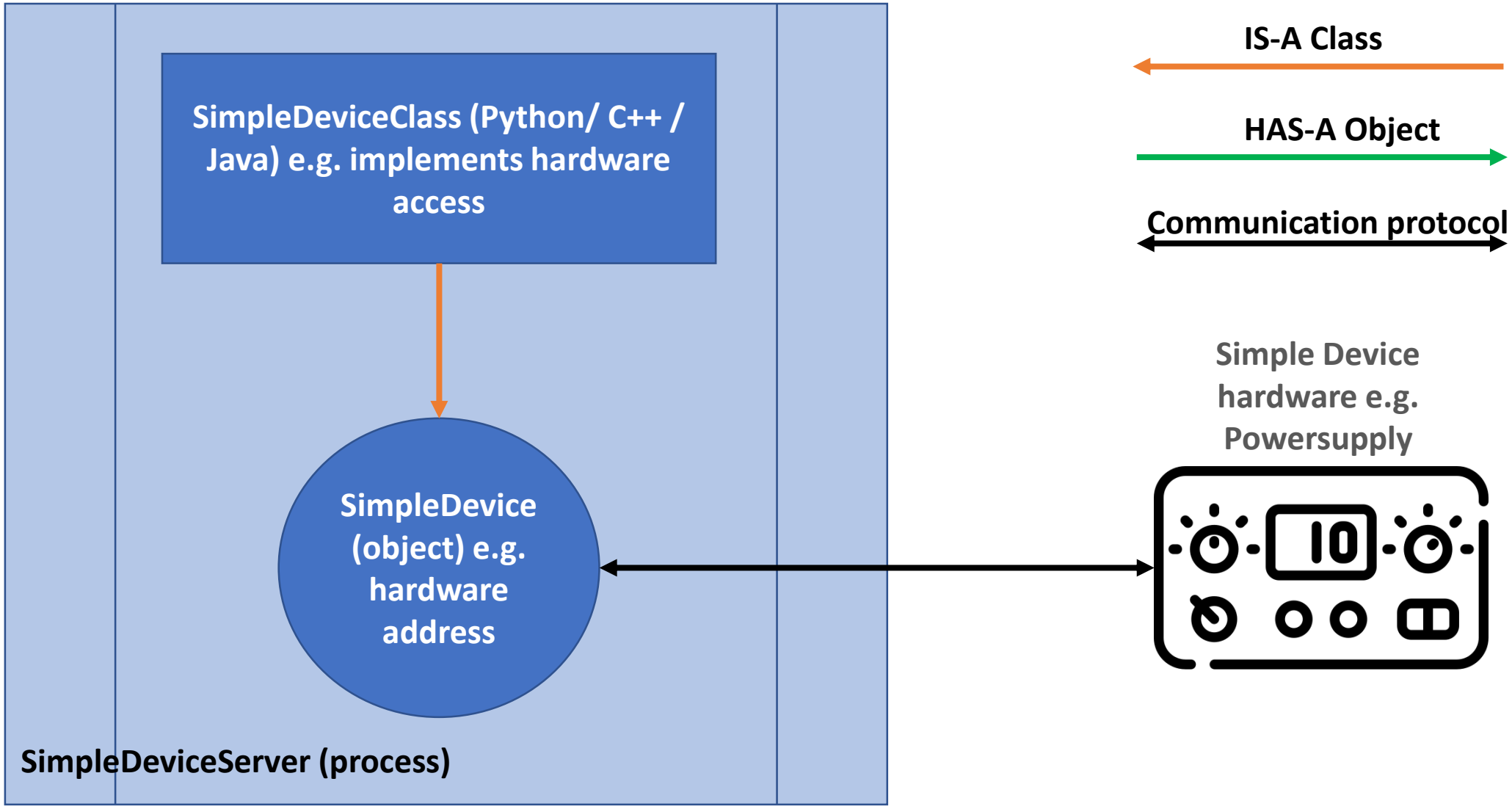
Forces / Constrains

- Keeping the **solution** to yourself doesn't require any effort
- Sharing the **solution** verbally helps a few others but won't make a big impact in your field.
- Writing down your understanding of the **solution** is hard work and requires much reflection on how you solve the **problem**.
- Transforming your specific **solution** into a more widely applicable **solution** is difficult.
- People are unlikely to use a **solution** if you don't explain the reasons for using it.
- Writing down the **solution** may compromise your competitive advantage (either personal or corporate.)

Solution

- Write down the **solution** using the pattern form. Capture both the **problem** and the **solution**, as well as the reasons why the **solution** is applicable. Apply *Mandatory Elements Present* to ensure that the necessary information is communicated clearly. Include *Optional Elements When Helpful* to capture any additional useful information. Distribute the resulting pattern to the largest audience you feel it could help that does not compromise your competitive advantage. Often, this means publishing your patterns exclusively within your company via Intranets or company journals.

Pattern #1 – SimpleDevice



Pattern #1 – SimpleDevice

TANGO
Connecting things together

1. **Context**

   need to remotely control some hardware in a lab and/or integrate into an existing control system to monitor it

2. **Problem**

   Equipment is in a remote location or needs to be accessed from another software language e.g. Python / Matlab, and controlled with

3. **Forces**

   Choose the right language

   Define Attributes, Commands and State machine

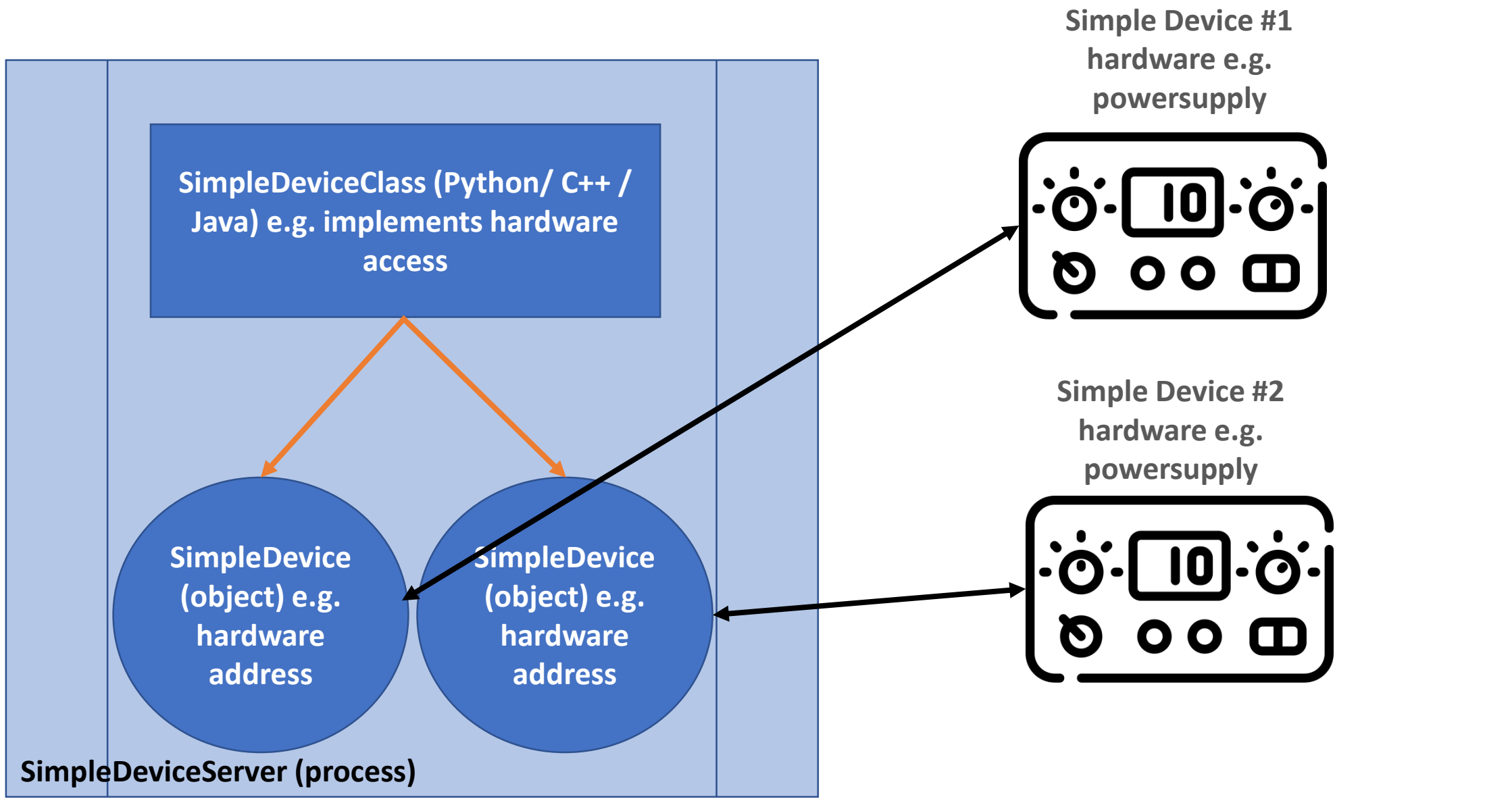
4. **Solution**

   Write a simple device server in Python/C++/Java

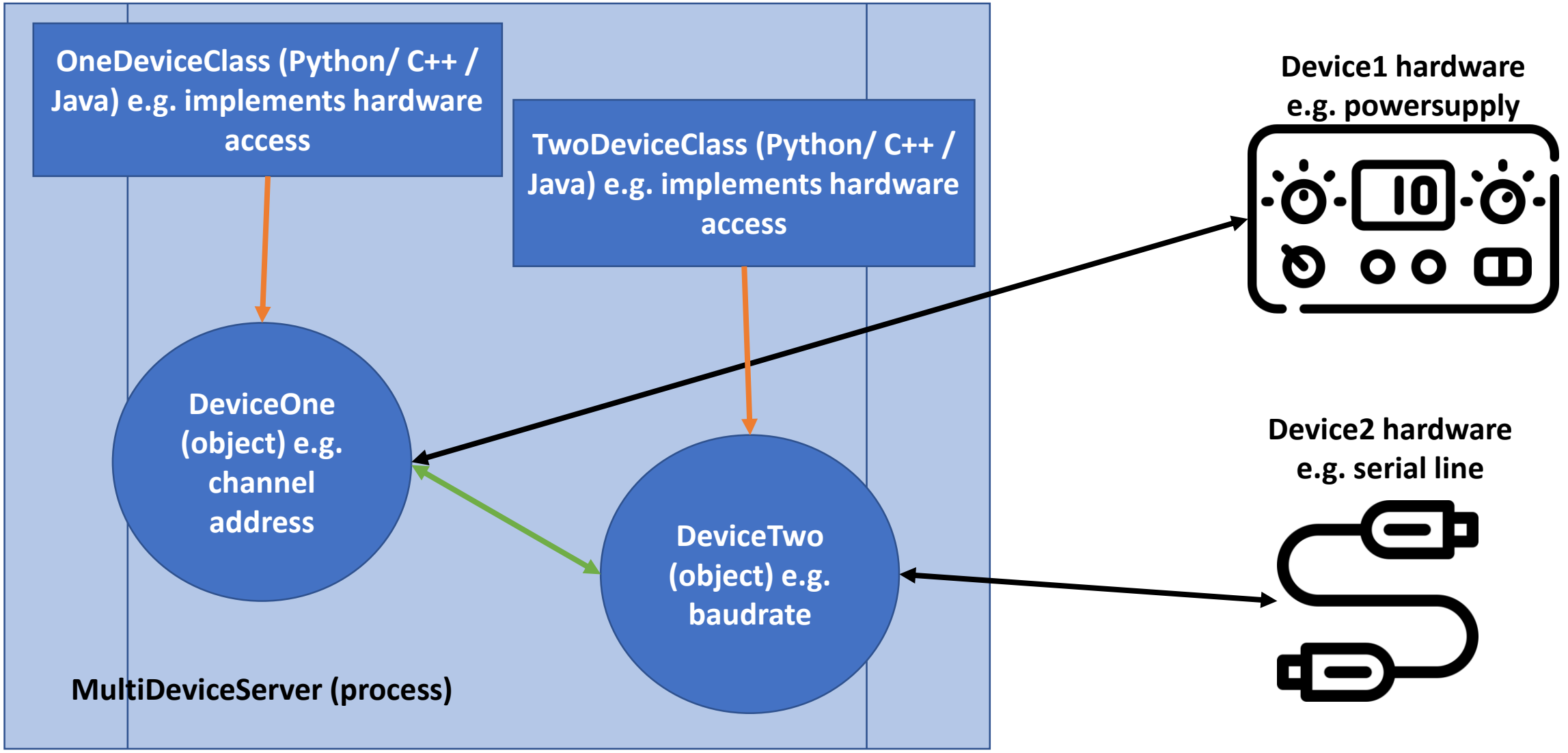
5. **Examples**

   PowerSupply, StepperMotor, VacuumGauge, Thermocouple, …

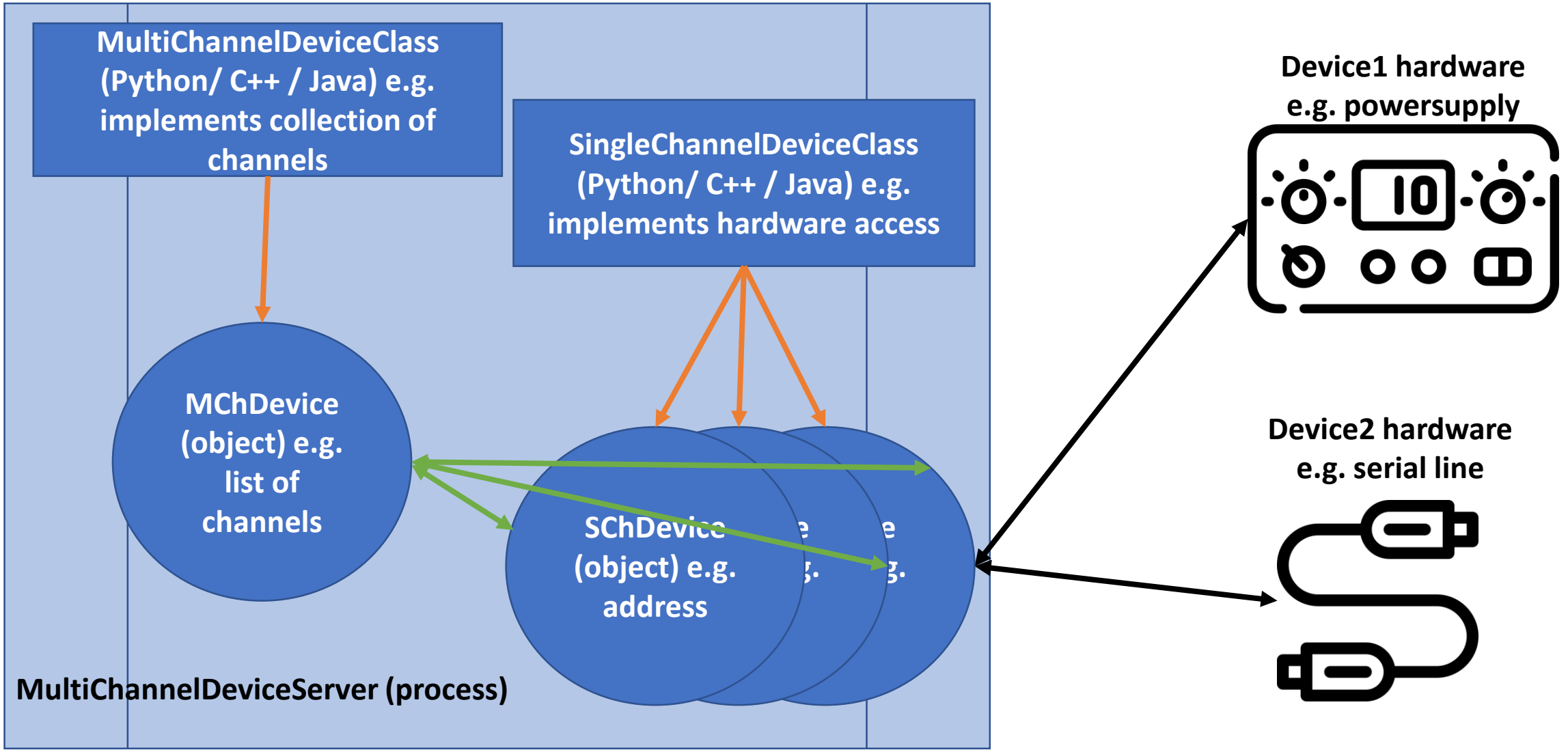
Pattern #2 – MultiSimpleDevice



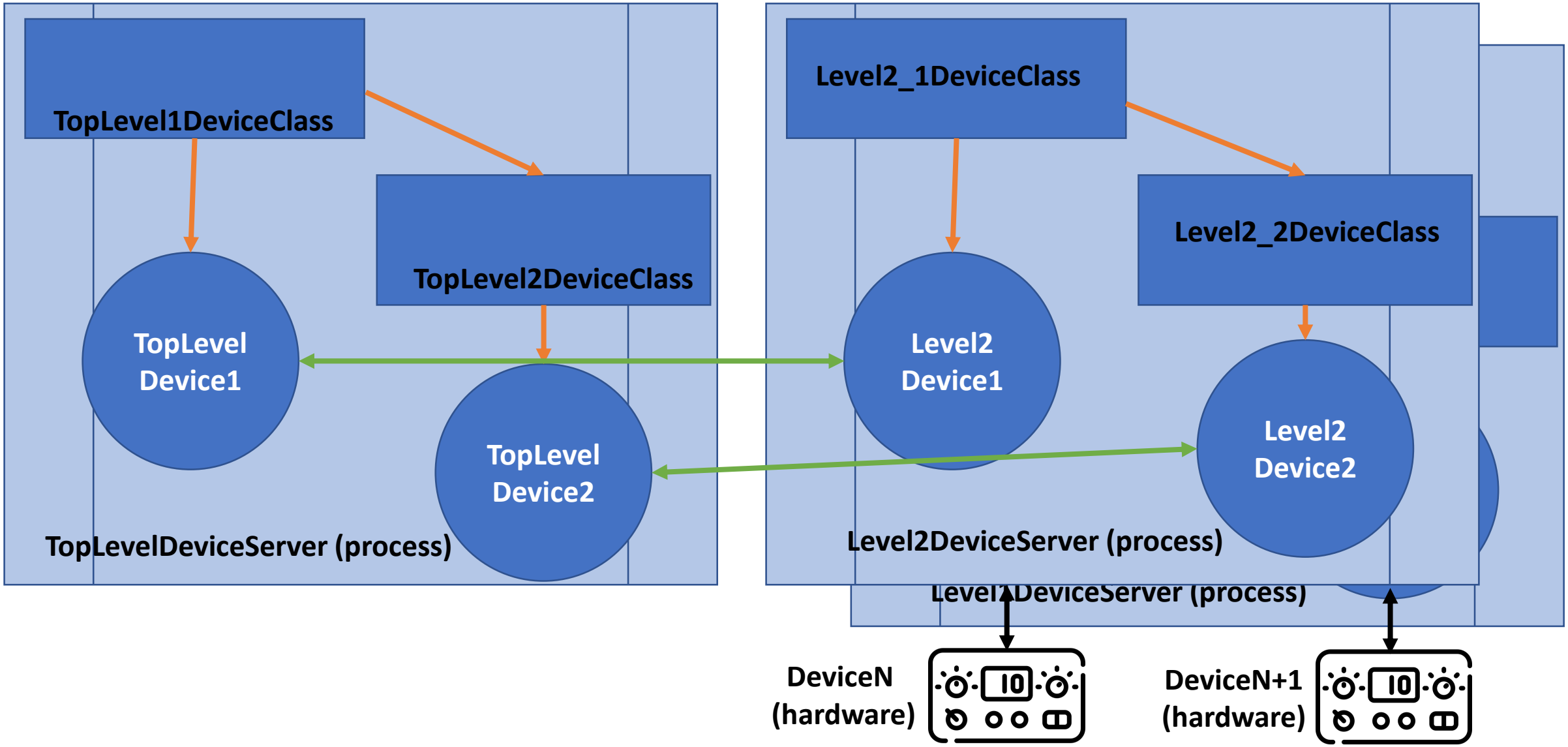
Pattern #3 – MultiDeviceClass



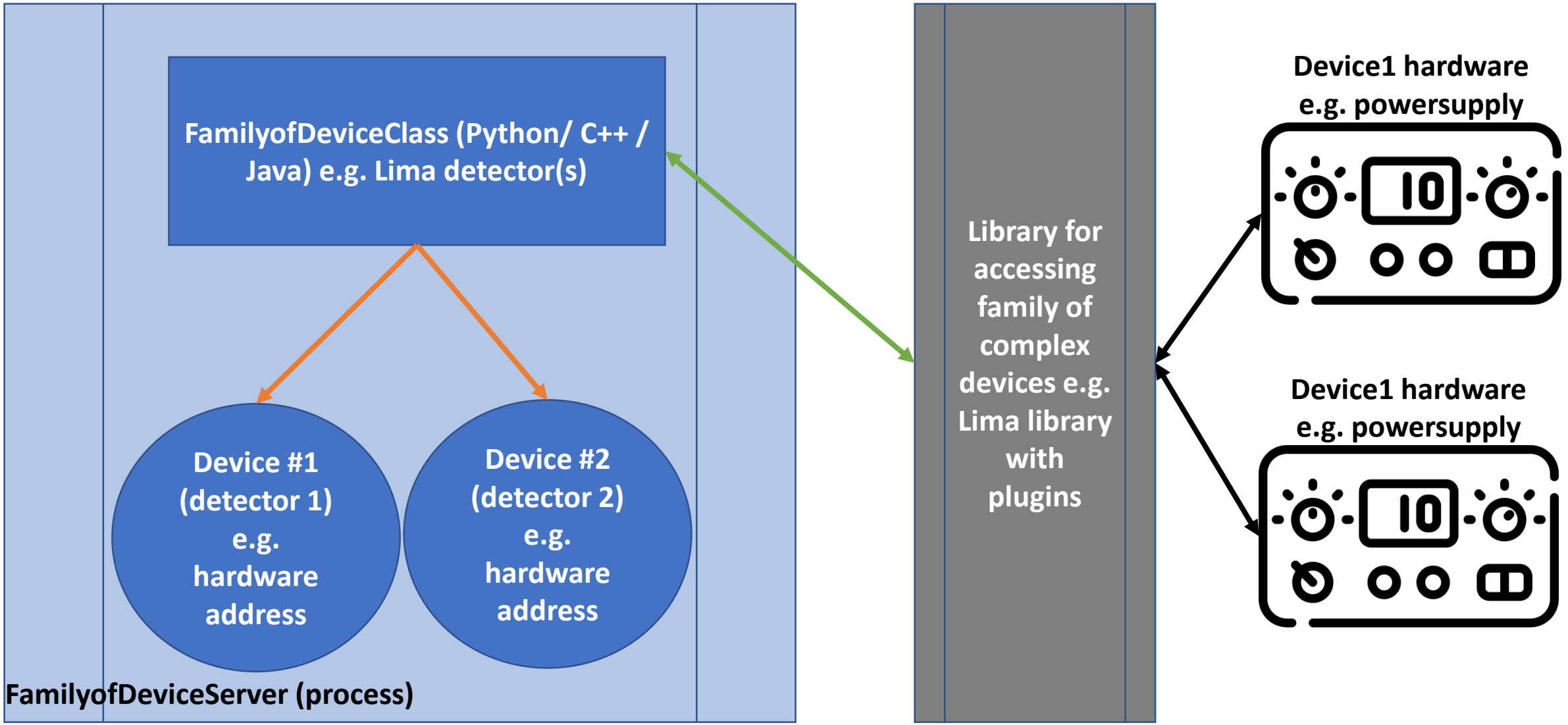
Pattern #4 – MultiChannelDeviceClass



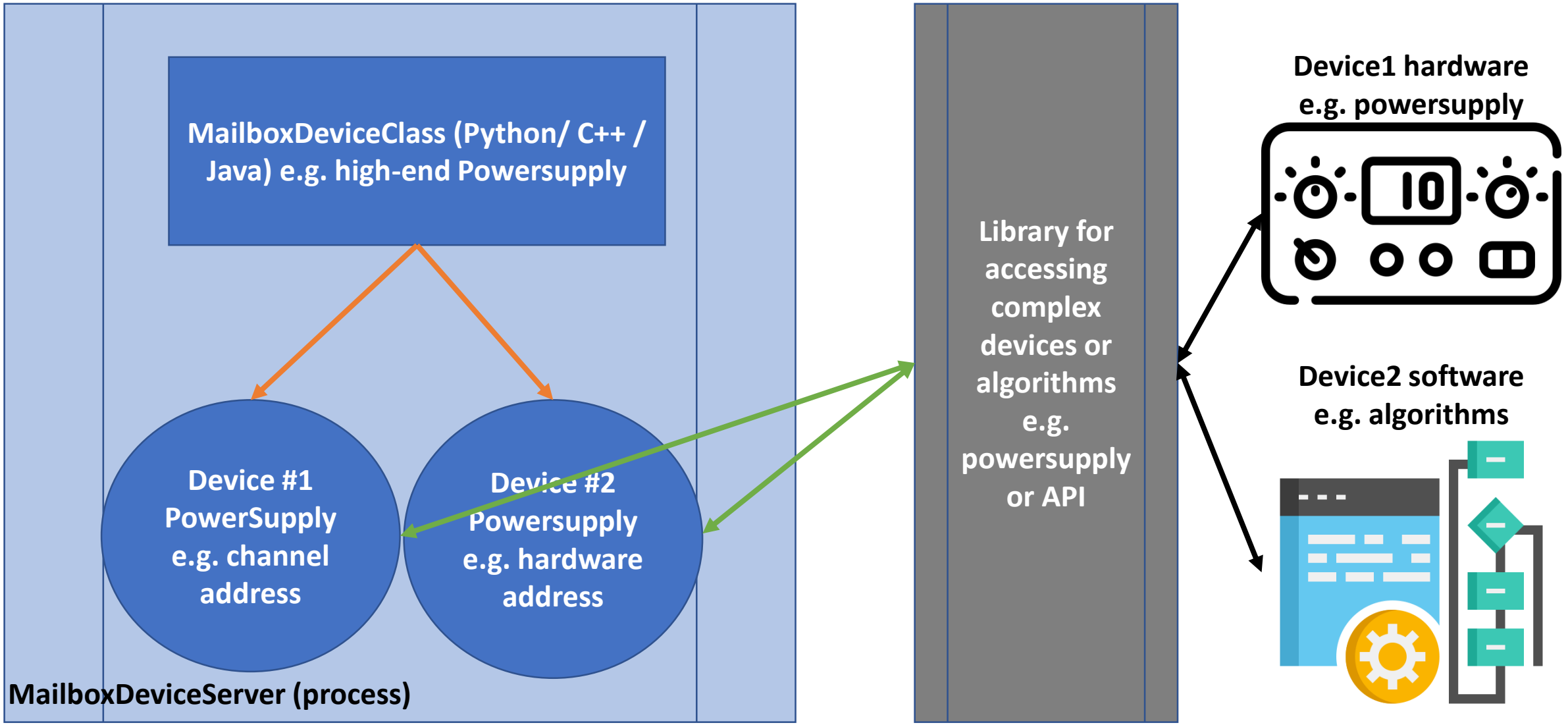
Pattern #5 – MultiLevelClass



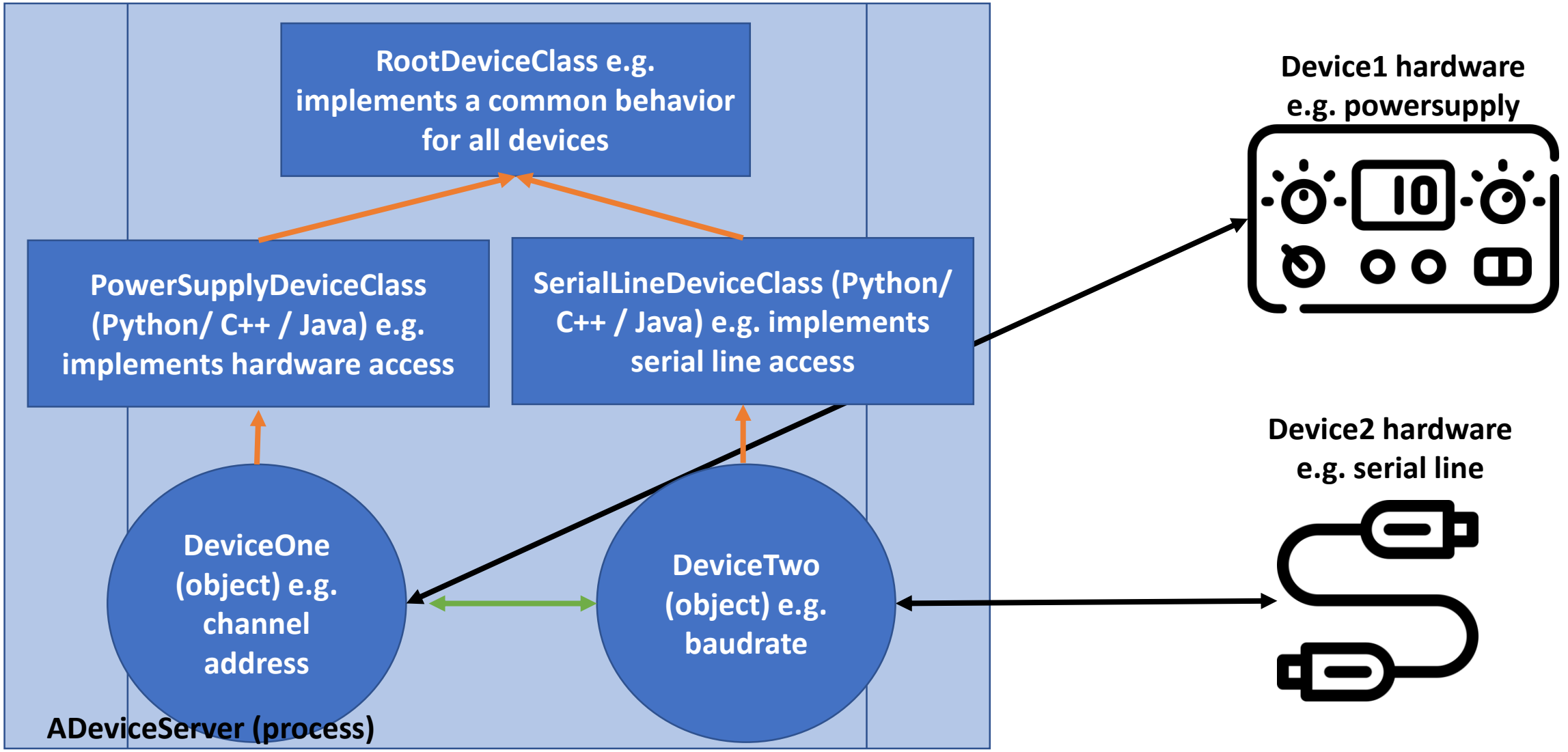
Pattern #6 – FamilyOfComplexDevice



Pattern #7 – MailboxDevice



Pattern #8 – RootDeviceClass



Anti-Patterns

1. **IgnoreState**
   Device has no state machine
   State is not implemented in the State attribute
2. **DeleteStaticAttributes**
   Static attributes are treated as dynamic attributes
3. **CreatorDestroyer**
   Device is created for very brief period (< 1s)
4. **AttributesAsCommands**
   Control points are implemented as Commands

Conclusion

1. There are many useful Tango Patterns in use at different sites

2. This talk has presented a few but would like to gather more from the community and extend them to more complex patterns / contexts

3. **Next steps**: include these and your patterns in the Tango documentation