

About the joy and tears of testing Embedded Devices

Chris Fiege <cfi@pengutronix.de>



<https://www.pengutronix.de>





About Pengutronix



- 30+ Colleagues
- Embedded Linux Operating System Development
- Customers in all Industries



About Me



Chris Fiege
Senior Hardware Developer

✉ cfi@pengutronix.de

🐦 [SmithChart](#)

🐙 [SmithChart](#)

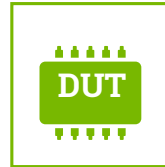


Agenda

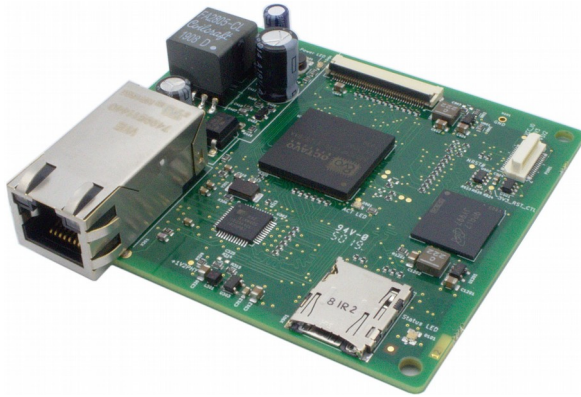
- 1) Why lab automation?
- 2) Labgrid overview
- 3) Demos!
- 4) Lessons learned



Controlling an Embedded Linux Device?



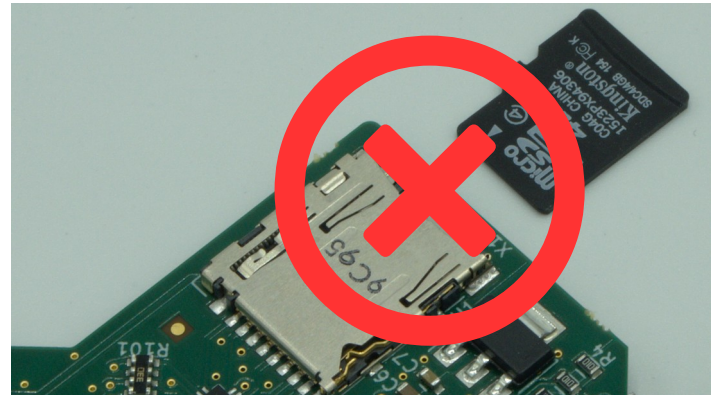
Motivation for Lab Automation?



© <https://www.needpix.com/photo/download/697794/>





CC BY-SA 2.0 https://commons.wikimedia.org/wiki/File:Series_of_build_lights.jpg



labgrid



- Python-Library
- Open Source: License: LGPL 2.1
-  github.com/labgrid-project/
-  labgrid.readthedocs.io



Design Criteria

- Shared hardware pool for interactive and CI/CT jobs



Design Criteria

- Shared hardware pool for interactive and CI/CT jobs
- No software components on the DUT



Design Criteria

- Shared hardware pool for interactive and CI/CT jobs
- No software components on the DUT
- Intended to be extendable:
Nothing should be in your way for special cases.



Design Criteria

- Shared hardware pool for interactive and CI/CT jobs
- No software components on the DUT
- Intended to be extendable:
Nothing should be in your way for special cases.
- No integrated scheduler
- No integrated build system



Design Criteria

- Shared hardware pool for interactive and CI/CT jobs
- No software components on the DUT
- Intended to be extendable:
Nothing should be in your way for special cases.
- No integrated scheduler
- No integrated build system
- No new testing framework



labgrid: Lab Hardware Abstraction

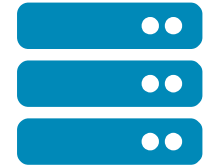
labgrid



Distributed Architecture: labgrid's View



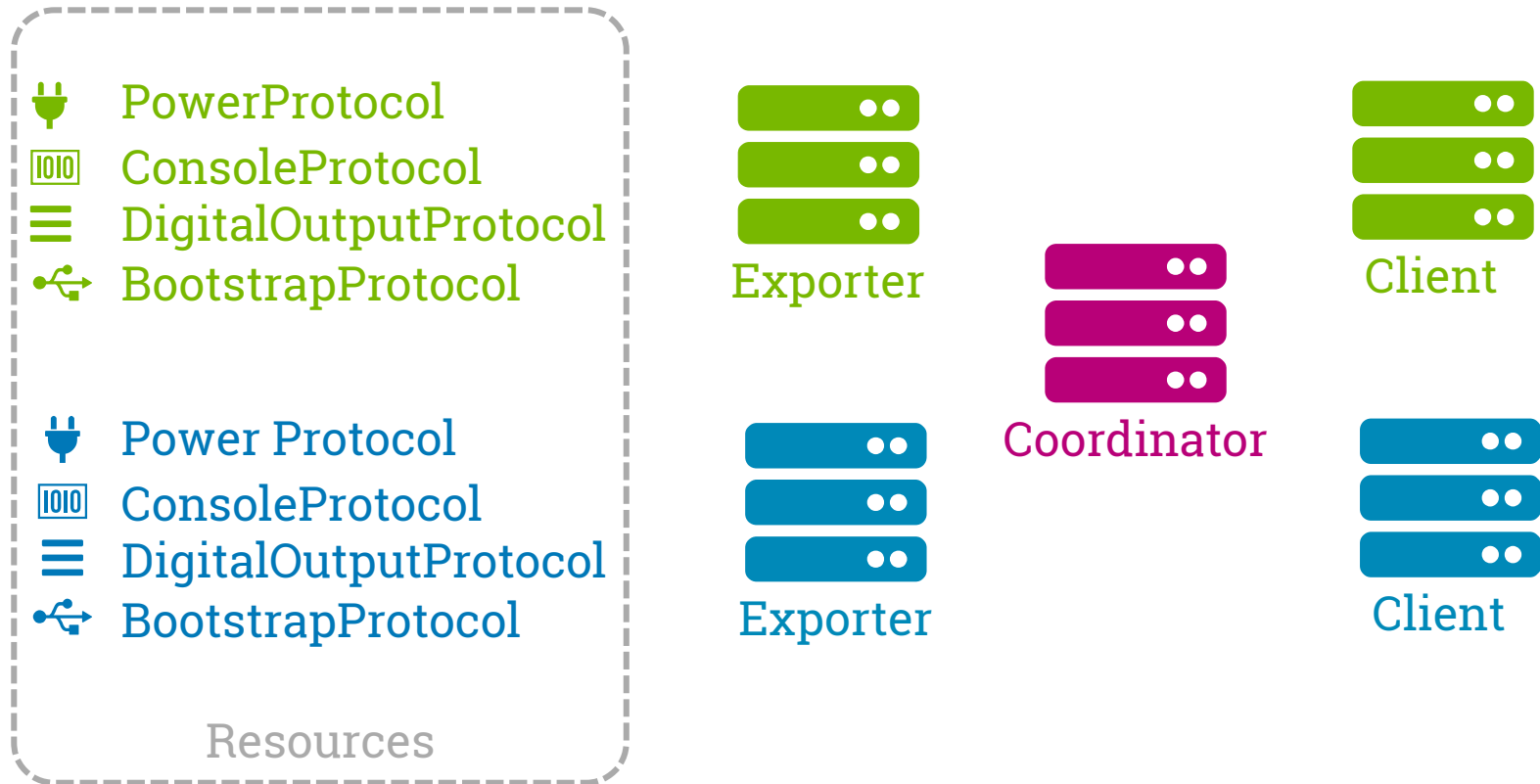
Client



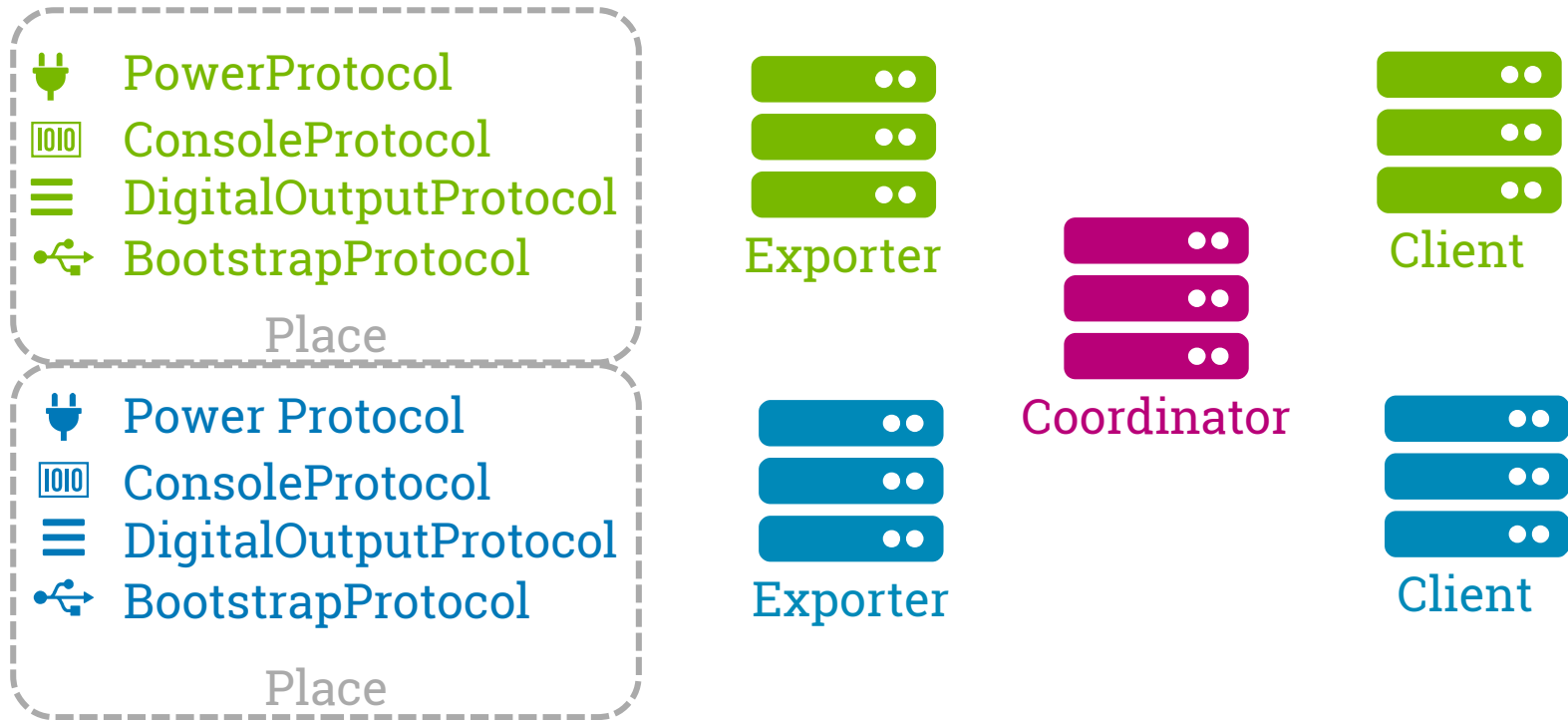
Client



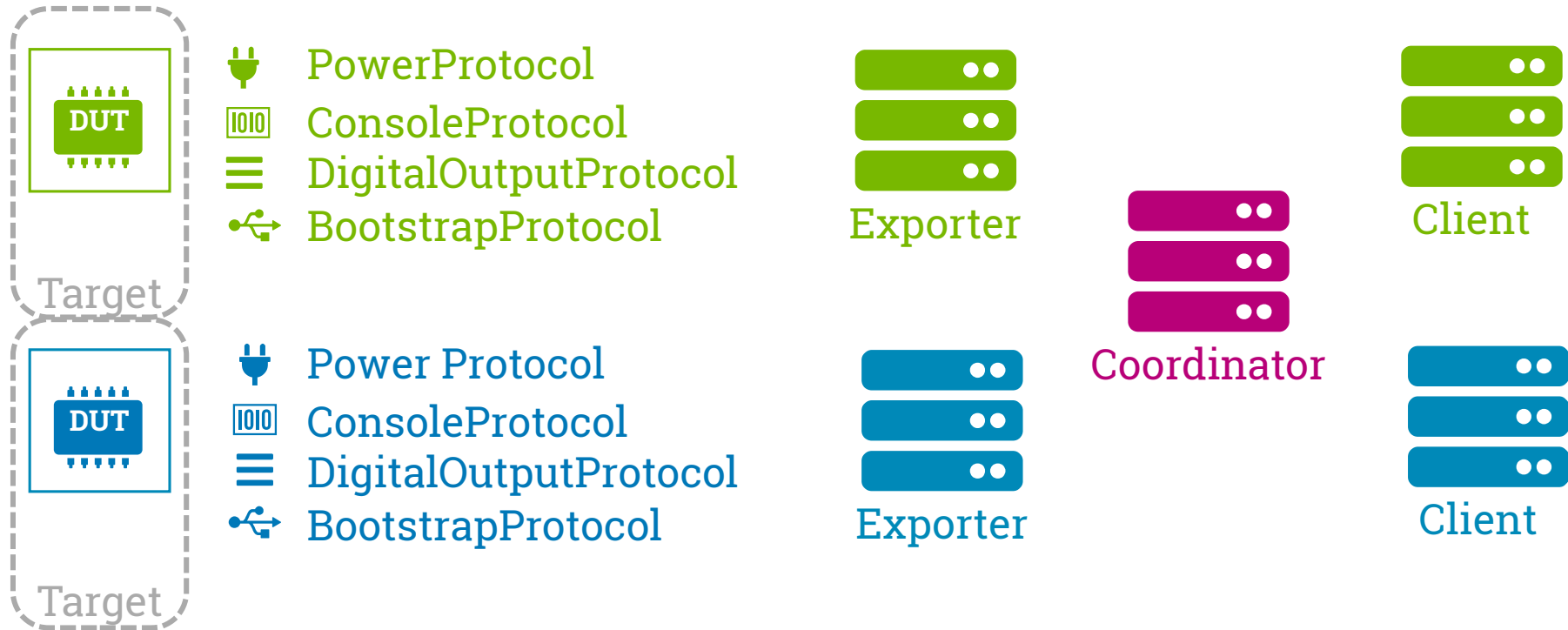
Distributed Architecture: labgrid's View



Distributed Architecture: labgrid's View



Distributed Architecture: labgrid's View



Demo Time



Lessons Learned

Single hardware pool for interactive and CI/CT



Lessons Learned

Single hardware pool for interactive and CI/CT

- ⊕ Only one set of hardware is needed
- ⊕ Easier debugging of failing tests



Lessons Learned

Single hardware pool for interactive and CI/CT

- ⊕ Only one set of hardware is needed
- ⊕ Easier debugging of failing tests
- ⊕/⊖ Added complexity for provisioning of DUT from scratch



Lessons Learned

Test suite has full hardware control



Lessons Learned

Test suite has full hardware control

- ⊕ Handling of any special or edge cases for a DUT
 - labgrid allows custom code in test suites



Lessons Learned

Test suite has full hardware control

- \oplus Handling of any special or edge cases for a DUT
 - labgrid allows custom code in test suites
- \ominus Added complexity for full hardware control



Lessons Learned

Strategies are also useable for interactive and scripting



Lessons Learned

Strategies are also useable for interactive and scripting

- ⊕ Full control over DUT state in interactive and scripting
- ⊕ Reproduceable workflows
 - → Simple handover within your team



Lessons Learned

USB



Lessons Learned

USB

- ⊕ easy to use, widely available



Lessons Learned

USB is a bad idea

- ⊕ easy to use, widely available
- ⊖ stability issues
- ⊖ consumer USB devices have bugs
- ⊖ hard to debug



Lessons Learned

USB is a bad idea

- ⊕ easy to use, widely available
- ⊖ stability issues
- ⊖ consumer USB devices have bugs
- ⊖ hard to debug

https://elinux.org/File:ATS2019-cfi-embedded-testing_handout.pdf

Lessons Learned

Only Locking + Reservation and no Scheduler



Lessons Learned

Only Locking + Reservation and no Scheduler

- \oplus CI already has a scheduler, no need to re-invent the wheel



Lessons Learned

Only Locking + Reservation and no Scheduler

- \oplus CI already has a scheduler, no need to re-invent the wheel
- \ominus Relies on developers to unlock their DUTs for CI/CT to run



Lessons Learned

Dynamic Resources



Lessons Learned

Dynamic Resources

- ⊕ Allows to sync a workflow to (USB) devices appearing



Lessons Learned

Distributed Architecture



Lessons Learned

Distributed Architecture

- ⊕ DUTs can be shared and accessed from everywhere
- ⊕ Noisy or large DUTs can be outside of your office



Lessons Learned

Distributed Architecture

- \oplus DUTs can be shared and accessed from everywhere
- \oplus Noisy or large DUTs can be outside of your office
- \ominus System is complex: more moving parts involved
- \ominus Error reporting in such system is hard:
Identifying the real cause for an error is still a manual debugging effort.



Thank you!

Questions?



Design Criteria

- Shared hardware pool for interactive and CI/CT jobs
- No software components on the DUT
- Expandable software architecture:
Nothing should be in your way for special cases.
- No integrated scheduler
- No integrated build system

