Designing Hardware-independent Testing Laboratory API

Paweł Wieczorek

August 22, 2019

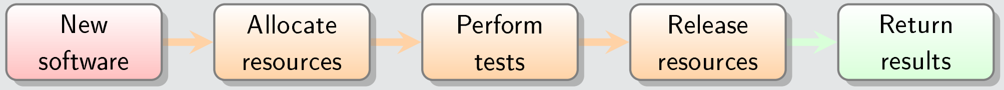
Samsung R&D Institute Poland

Outline

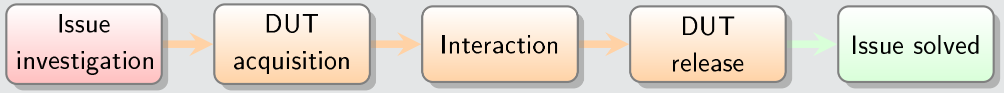
Motivation

Use cases

Automated testing



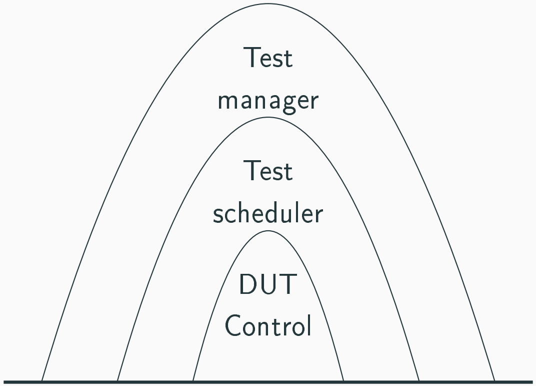
Direct access (*hacking*)



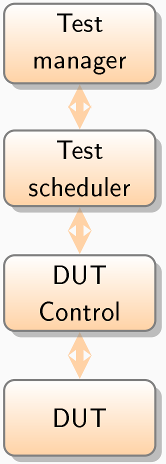
Abstractions (https://elinux.org/Test_Standards)

TM	Test Manager: Actions initiation
TS	Test Scheduler: Resource allocation, scheduling
DUT-C	DUT Control: Controlling power, providing network, ensuring communication, grabbing logs
DUT	Device Under Test

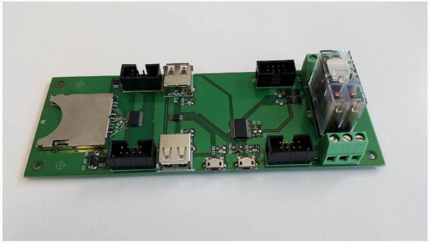
Common laboratory structure

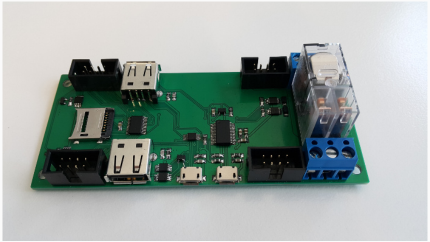


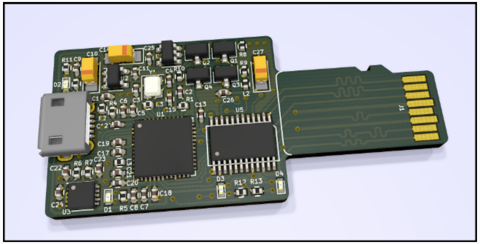
Decoupled laboratory structure

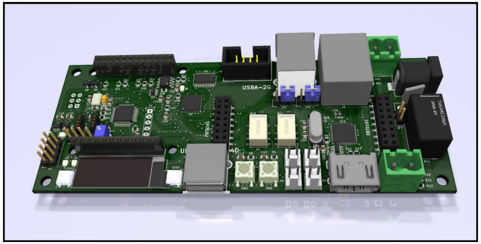


Devices

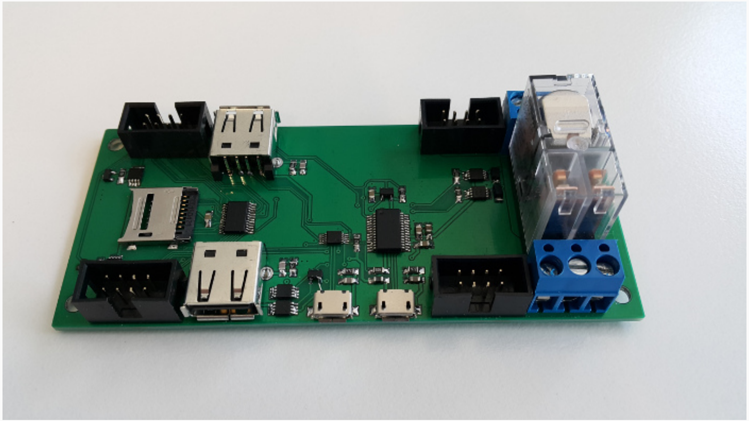






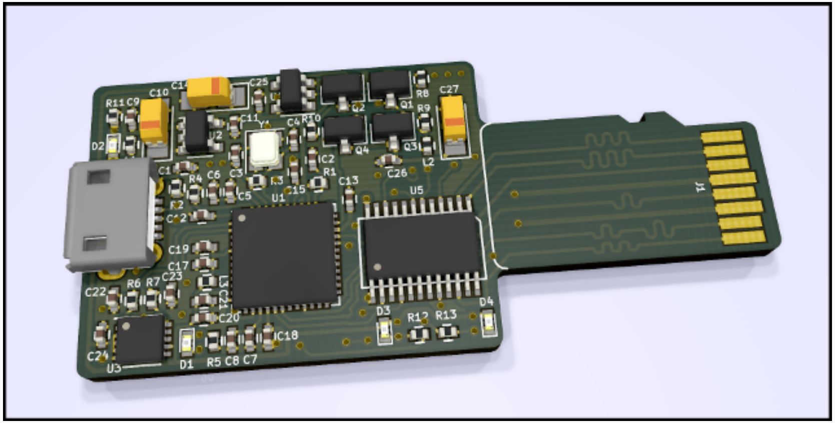


SD-MUX



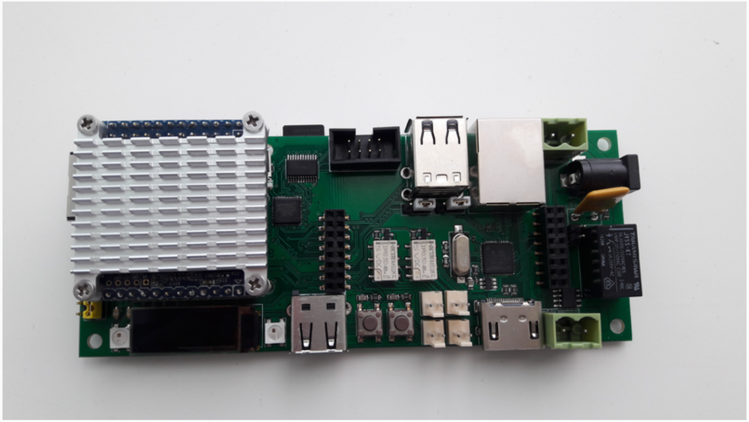
https://wiki.tizen.org/SD_MUX

SDWire



https://wiki.tizen.org/SDWire

MuxPi



https://wiki.tizen.org/MuxPi

Testing laboratory layers

Challenges

Knowledge

Which actions are necessary? Where can it be performed? How to do it?

Responsibilities

Who performs given action?

Sharing

Who can use DUT? How can DUT be used?

Implementation

Test Manager (minimal)

- initiate actions
- list (or cancel) currently performed actions

Test scheduler (generic)

- list available resources, request specific ones
- acquire assigned resources (then prolong, finally release)

DUT Control (*tricky*)

- boot (and login)
- execute commands
- copy files

Case study

Test Manager

Strengths

- Requires only preparing test plan
- Test plans can be reused among various projects

Weaknesses

- Keeping compliance
- Catching up with others (e.g. LAVA, SQUAD)

Test Scheduler

Strengths

- Users treated equally
- Resource type-agnostic

Weaknesses

- Requires additional agent
- Capabilities declared up front

DUT Control

Strengths

- Only some knowledge required
- Unification possibility

Weaknesses

- Hard initial setup
- Often unique for a given testing laboratory

Summary

Outcome

- Unable to demo without specific hardware
- Risky large scale deployments
- Responsibilities division allows easier onboarding

Conclusion

- User-centric approach resulted in smaller building blocks
- Smaller blocks could be easier swapped or used independently
- Improvement needs more *reuse* instead of *rewrite*

`https://github.com/SamsungSLAV`

Thank you!

Paweł Wieczorek
p.wieczorek2@samsung.com