

# Embedded Security Testing

Rainer Poisel

# Overview

- Introduction (10 Min.)
- Practical Example (10 Min.)
- Advanced Topics (Remainder)
- Q&A (remainder)

# About the Speaker

- Rainer Poisel
  - Software Development and Architecture
  - Embedded CI/CD
  - Cyber Security of IACS

# Embedded Systems

Specialized computers within larger systems.

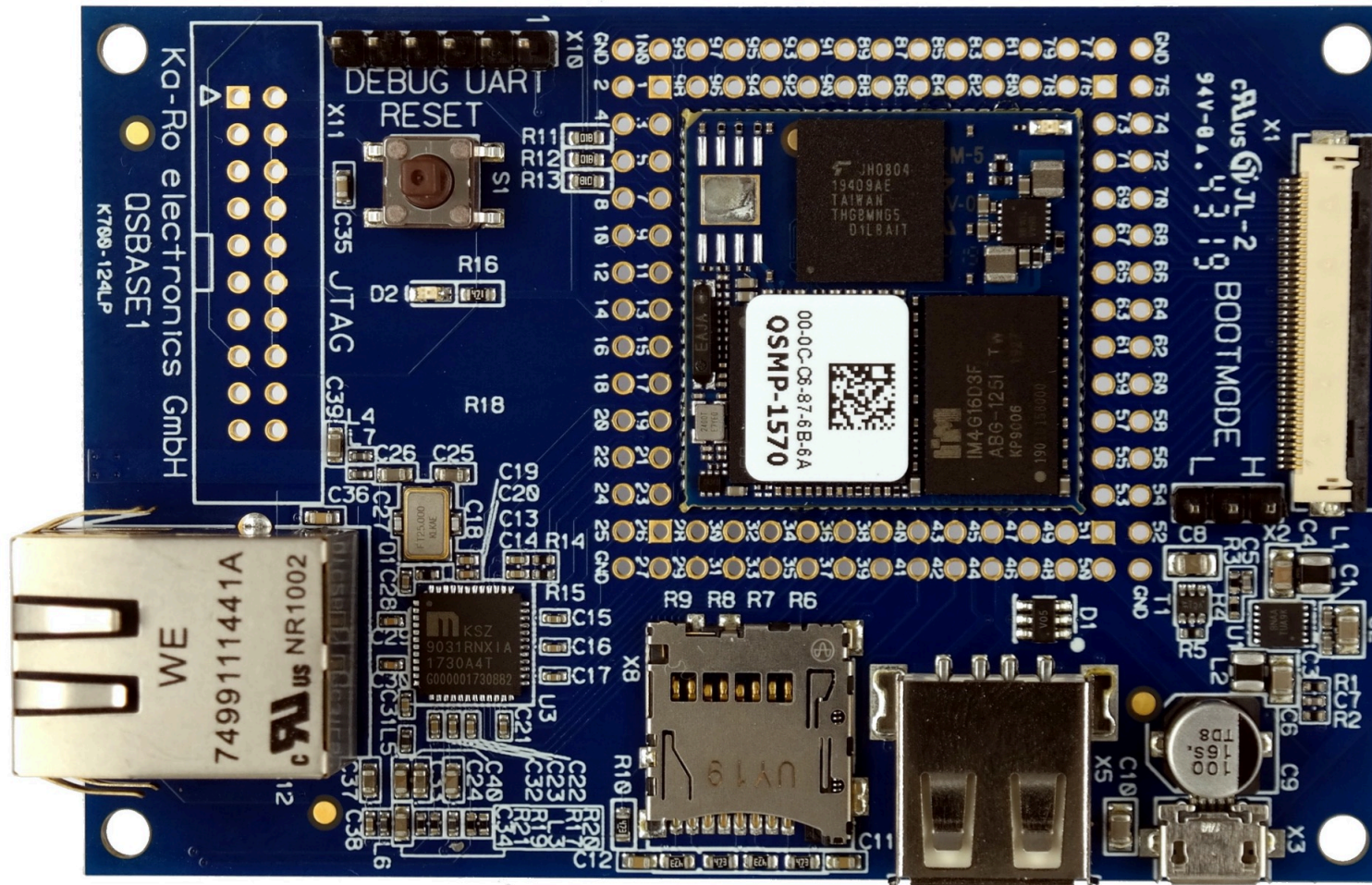


Figure 1: STM32MP1 based SBC

# IACS

## Industrial Automation and Control Systems / PLCs



Figure 2: Programmable Logical Controllers  
A By Pierre75000 - Own work, CC BY-SA 4.0

# Motivation

# Customers are (not) the best software testers.



# Verifying Security

- International Standards: IEC 62443, (ISO/IEC 27001)
- Regulatory Frameworks:
  - Cyber Resilience Act
  - NIS 2 Directive
  - NIST SP 800-82
  - ...

# Guidelines

1. Verify that embedded devices are secure.
2. Keeping embedded tests simple.
3. Tests should support the development process.

# Keeping Tests Simple (I)

**Task:** Check that `uname -s` returns the string `Linux`

```
1 from labgrid.util.ssh import SSHConnection
2
3 def test_uname(ssh_cmd: SSHConnection) -> None:
4     stdout = ssh_cmd.run_check("uname -s")
5     assert stdout == "Linux"
```

Figure 3: Simple pytest/labgrid sample

# Keeping Tests Simple (II)

Not in the code:

- Power on
- Pass bootloader
- Wait for login
- Wait for shell
- Configure DHCP client (if required)
- Determine IP

# Supporting the CI / CD process

Continuous Integration and Deployment

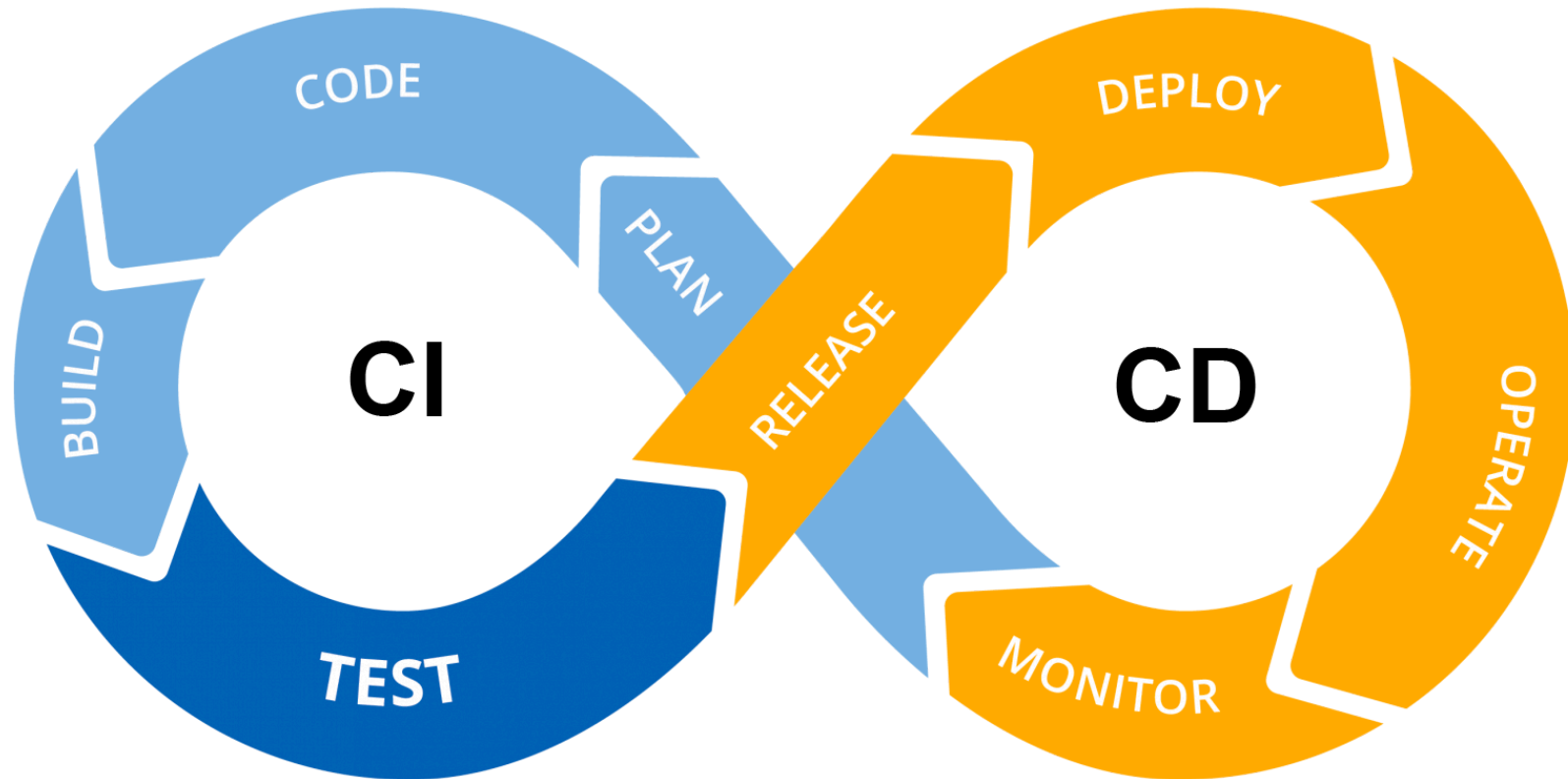


Figure 4: CI / CD Workflow  
padok.fr

# Intro to pytest/labgrid

# Software Testing

Evaluate and verify that systems meet requirements.

- Unit Tests
  - All dependencies are mocked.
- Integration Tests
  - Some dependencies are mocked.
- System Tests
  - Nothing is mocked.

# pytest

- pytest is a **testing framework**
- pytest is implemented in Python
- Provides:
  - Simple syntax for writing tests
  - Support for fixtures and plugins
  - Integration with continuous integration tools



# pytest: Simple Example (I)

```
1 def inc(x: int) -> int:  
2     """Function under test."""  
3     return x + 1  
4  
5  
6 def test_answer() -> None:  
7     assert inc(3) == 4
```

Figure 5: Simple Example: Correct Implementation

# pytest: Simple Example (II)

```
1 def double(x: int) -> int:
2     """Function under test."""
3     return x + 2 # oops, typo
4
5
6 def test_answer() -> None:
7     assert double(3) == 6 # fails, 3 + 2 != 6
```

Figure 6: Simple Example: Typo in Implementation

# labgrid (I)

- labgrid is **NOT** a testing framework
- labgrid is implemented in Python
- Provides:
  - Hardware-Abstraction
  - DUT pool management
  - Remote control
  - pytest integration



# labgrid (II)

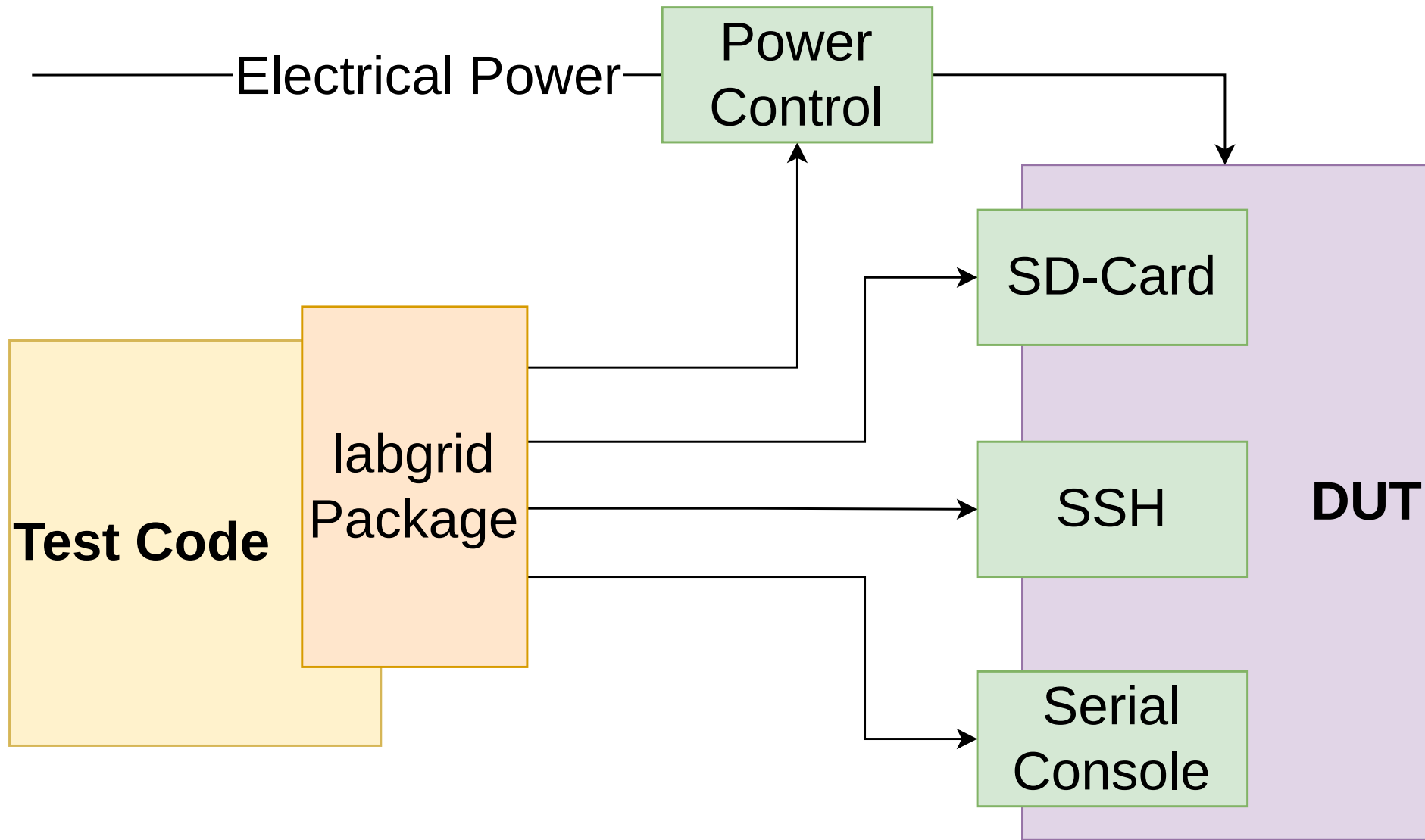


Figure 7: Using the Labgrid Library

# labgrid (III)

Use cases:

- Remote SD-Card access
- Data transfer (scp, sshfs, rsync, ...)
- Bootloader interaction
- Console access
- Network configuration
- GPIO access
- Power control
- ...

# labgrid (IV)

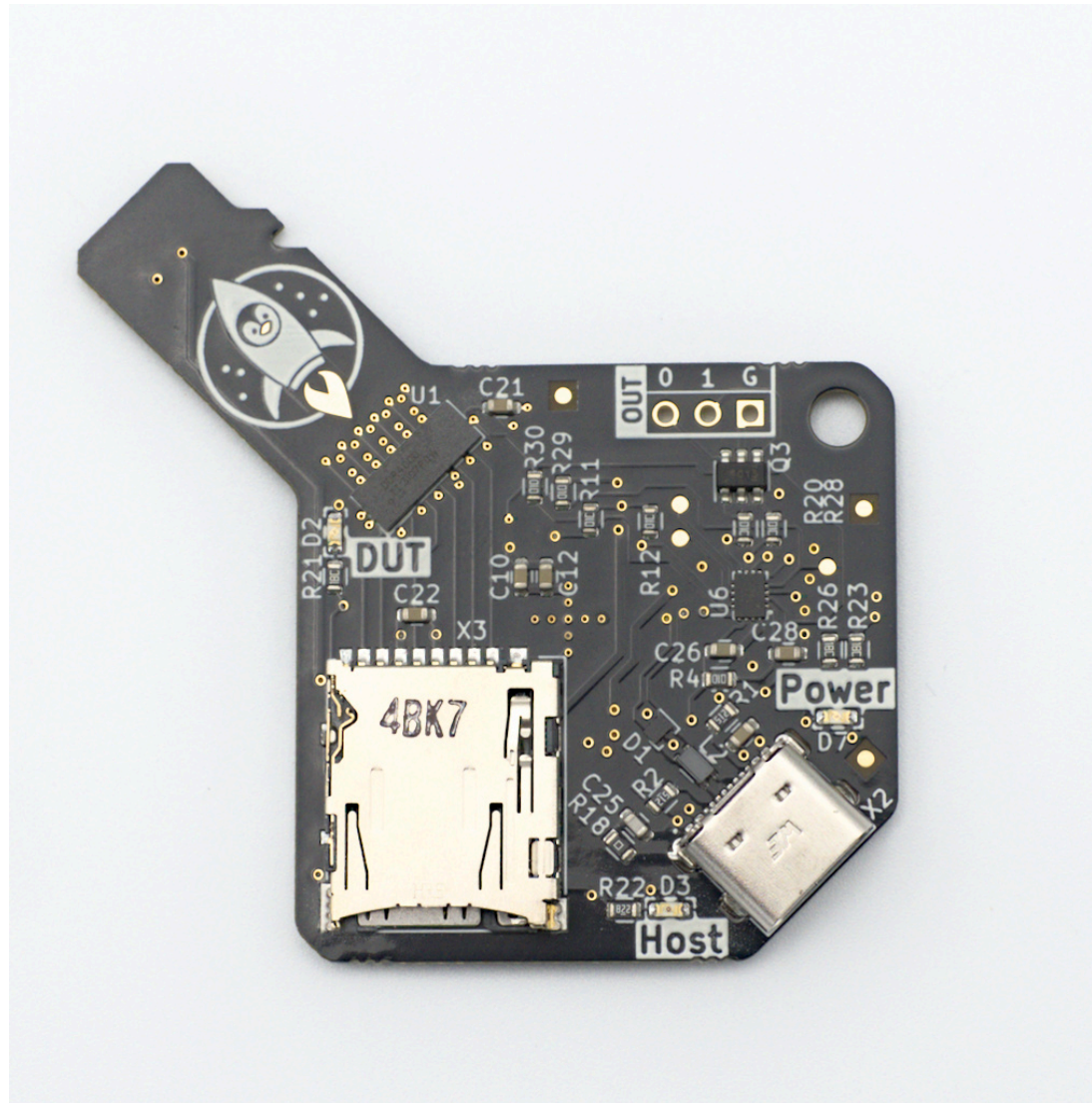


Figure 8: USB-SD-Mux FAST

# Practical Example

**Task:** Test the OpenVPN client of the OpenWrt Distribution



# Functional Requirements

Functional:

- **Connection Establishment**
- **Traffic Routing**
- Authentication Mechanisms
- Reconnection Attempts
- Configuration Loading
- DNS Configuration
- Logging
- ...

# Non-Functional Requirements

- Connection Time
- Resource Consumption
- Scalability
- Network Resilience
- Security
- Compatibility
- ...

# Architecture

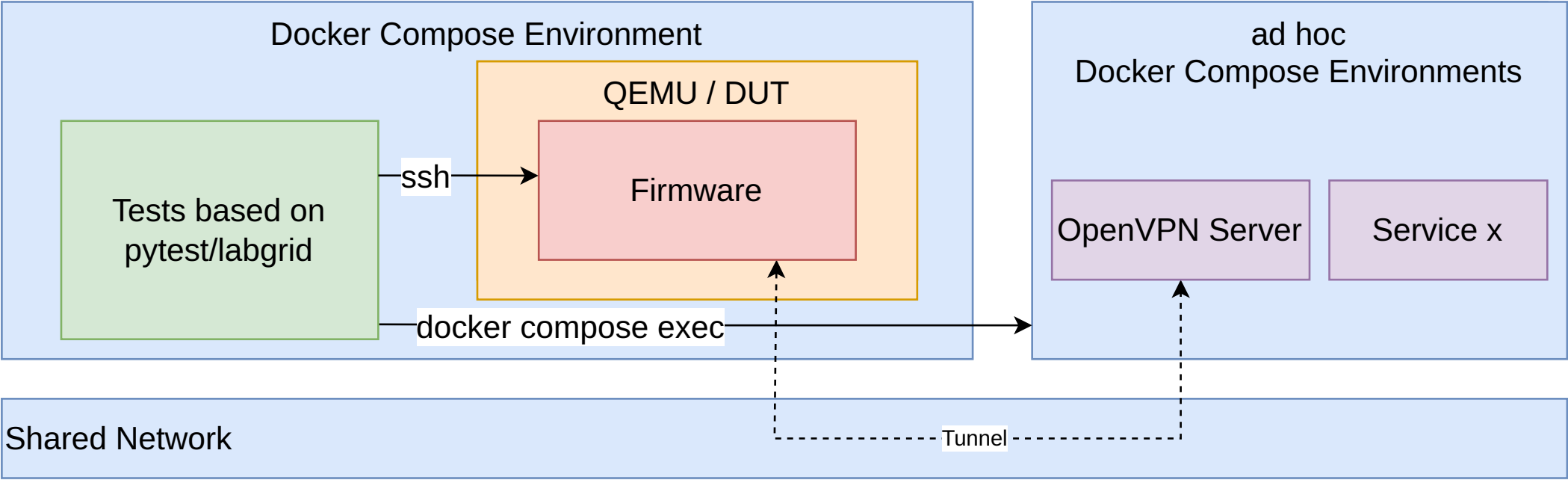


Figure 9: Labgrid QEMU Sample

# Demo





# Code Stats

## Logic:

- ~400 lines utility code
  - (PKI, docker compose, etc.)
- ~100 lines test code

## Configuration:

- ~90 lines YAML

# Advanced Topics

# Distributed labgrid (I)

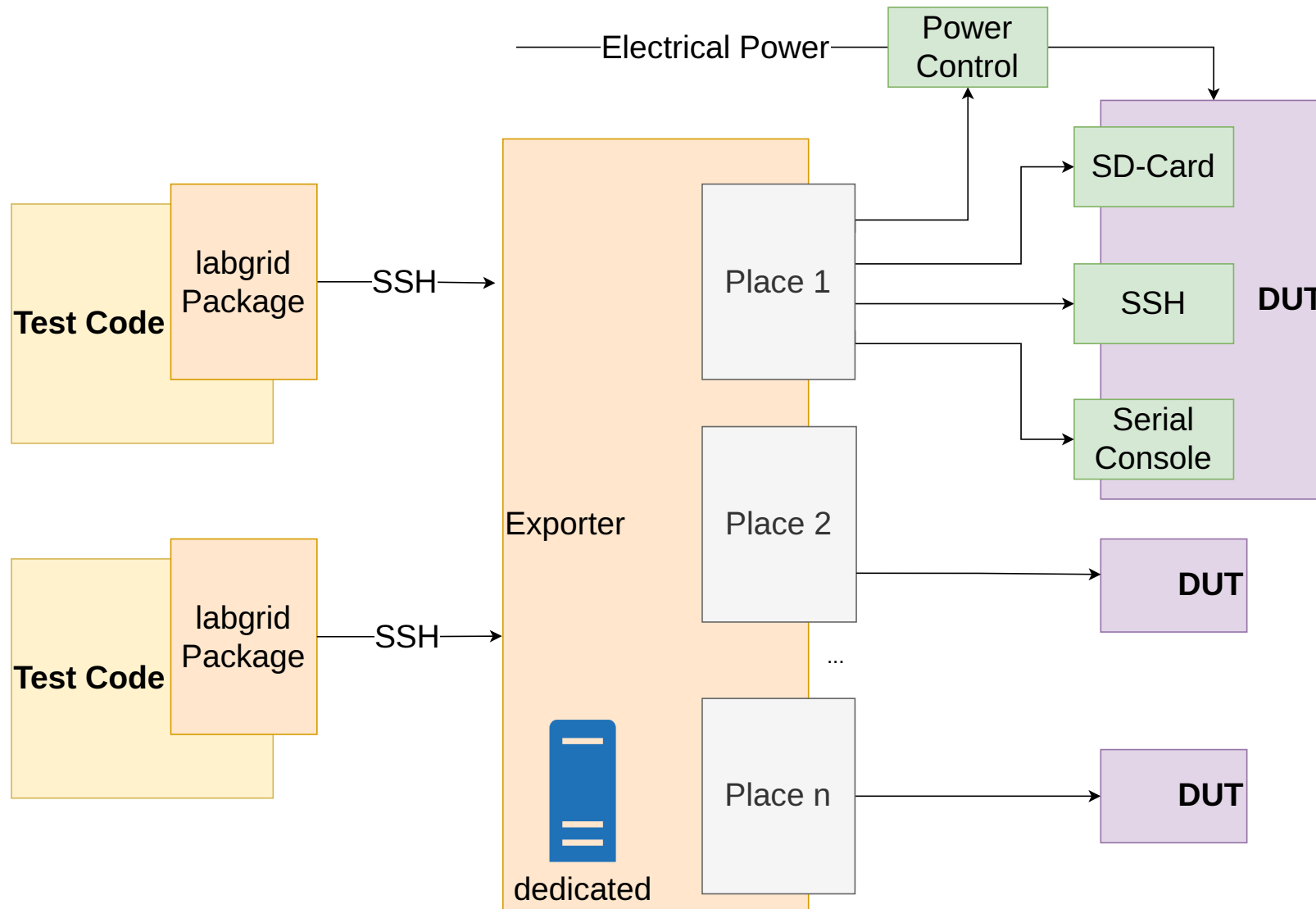


Figure 10: Distributed Labgrid Architecture

# Distributed labgrid (II)

## Use Cases:

- Device Pooling
  - (Remote) Developers
  - CI/CD
- Parallel Test Execution

# Addendum

# References

## **pytest**

<https://pytest.org>

## **labgrid**

<https://pengutronix.de/en/software/labgrid.html>

## **Sample Repository**

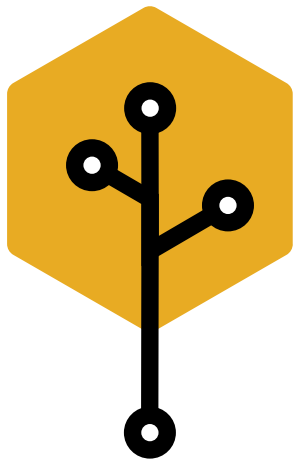
<https://github.com/honeytreelabs/labgrid-qemu-sample>

## **IEC 62443 vs. CRA Gap Analysis**

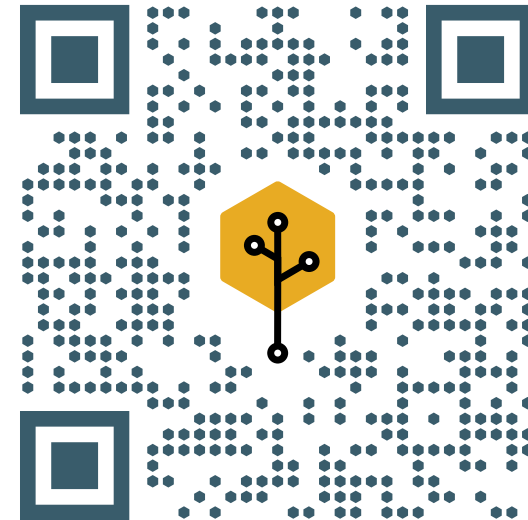
[https://honeytreelabs.com/posts/iec62443\\_vs\\_cra/](https://honeytreelabs.com/posts/iec62443_vs_cra/)

# Our Services

- Embedded SW Development
- Continuous Integration
- Continuous Development
- IIoT Security
- RF Protocols
- Reverse Engineering



**honeytree**  
Labs





# Backup Slides

# pytest: Fixtures (I)

```
1 import pytest
2 import smtplib
3
4
5 def test_ehlo() -> None:
6     smtp_connection = smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
7     response, msg = smtp_connection.ehlo()
8     assert response == 250
9     assert b"smtp.gmail.com" in msg
10
11
12 def test_noop() -> None:
13     smtp_connection = smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
14     response, msg = smtp_connection.noop()
15     assert response == 250
```

# pytest: Fixtures (II)

```
1 import pytest
2 import smtplib
3
4
5 @pytest.fixture(scope="module")
6 def smtp_connection() -> smtplib.SMTP:
7     return smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
```

Figure 11: Fixture Definition

```
1 def test_ehlo(smtp_connection: smtplib.SMTP) -> None:
2     response, msg = smtp_connection.ehlo()
3     assert response == 250
4     assert b"smtp.gmail.com" in msg
5
6
7 def test_noop(smtp_connection: smtplib.SMTP) -> None:
8     response, msg = smtp_connection.noop()
9     assert response == 250
```

Figure 12: Fixture Verwendung