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.

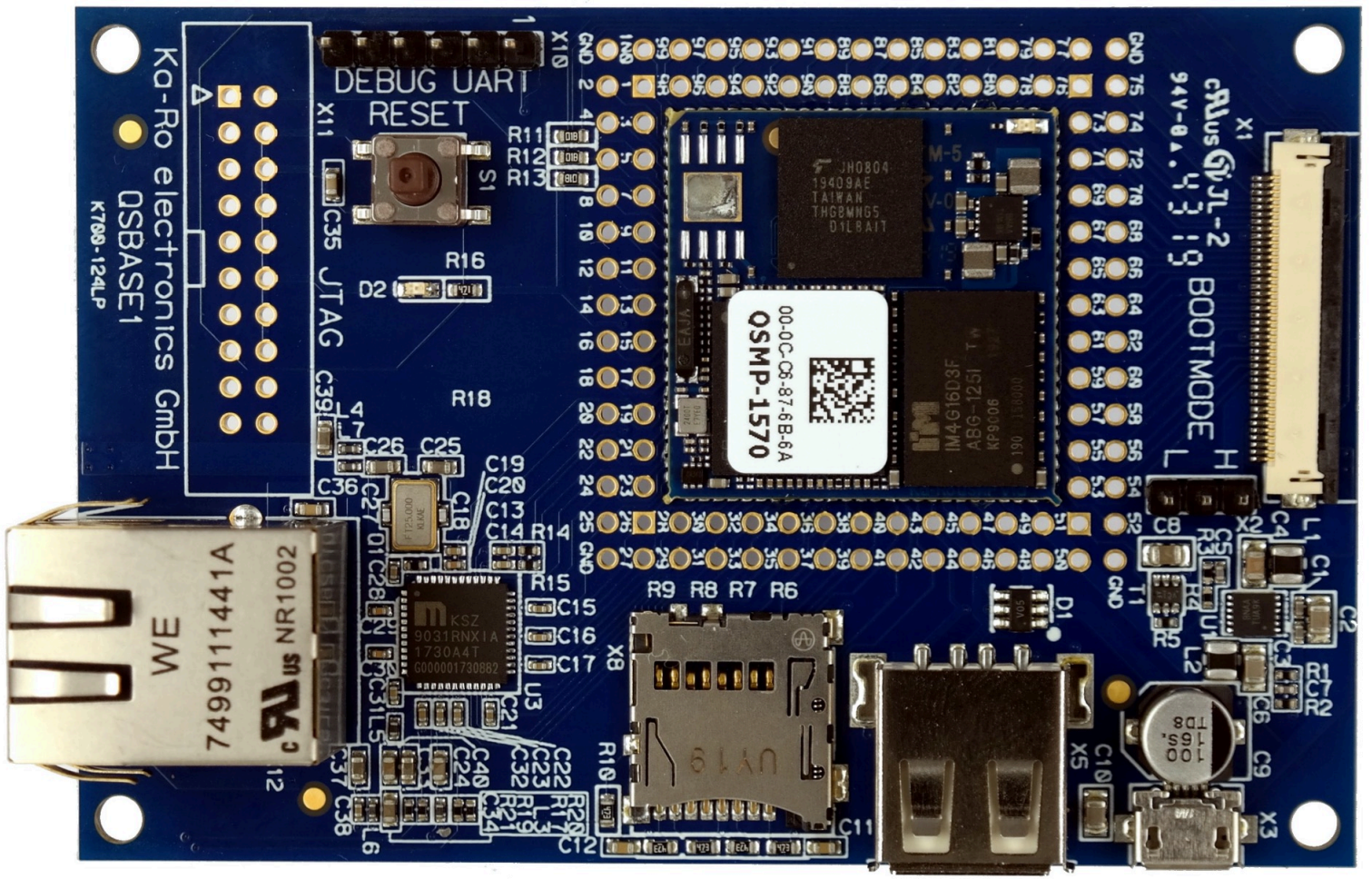

Figure 1: STM32MP1 based SBC

honeytree Labs

IACS

Industrial Automation and Control Systems / PLCs

Figure 2: Programmable Logical Controllers

A By Pierre75000 - Own work, CC BY-SA 4.0

honeytree Labs

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`

```python
from labgrid.util.ssh import SSHConnection

def test_uname(ssh_cmd: SSHConnection) -> None:
    stdout = ssh_cmd.run_check("uname -s")
    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

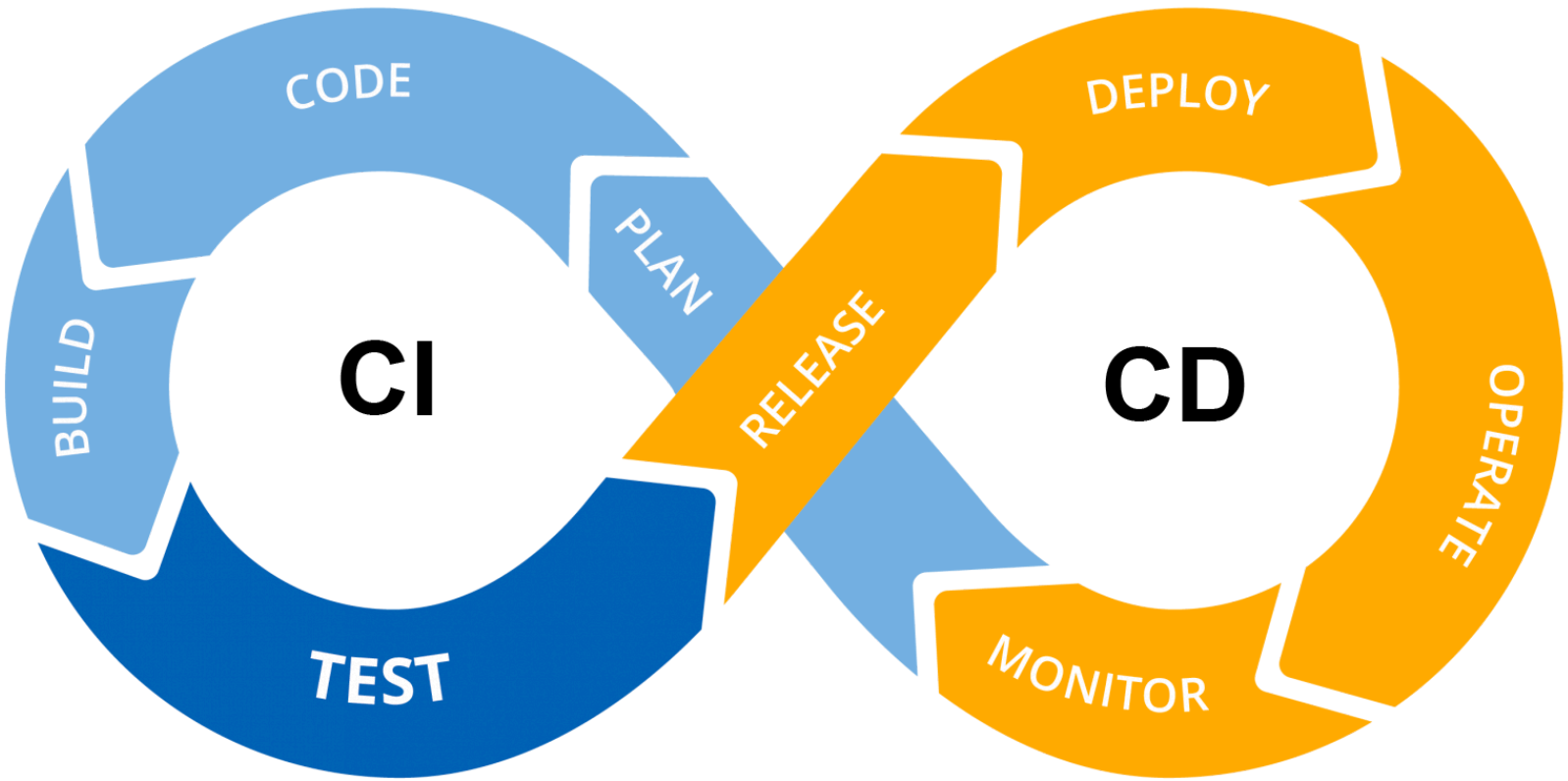

Figure 4: CI / CD Workflow

padok.fr

honeytree Labs

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

pytest: Simple Example (I)

```python
def inc(x: int) -> int:
    """Function under test."""
    return x + 1


def test_answer() -> None:
    assert inc(3) == 4
```

Figure 5: Simple Example: Correct Implementation

pytest: Simple Example (II)

```python
def double(x: int) -> int:
    """Function under test."""
    return x + 2  # oops, typo


def test_answer() -> None:
    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)

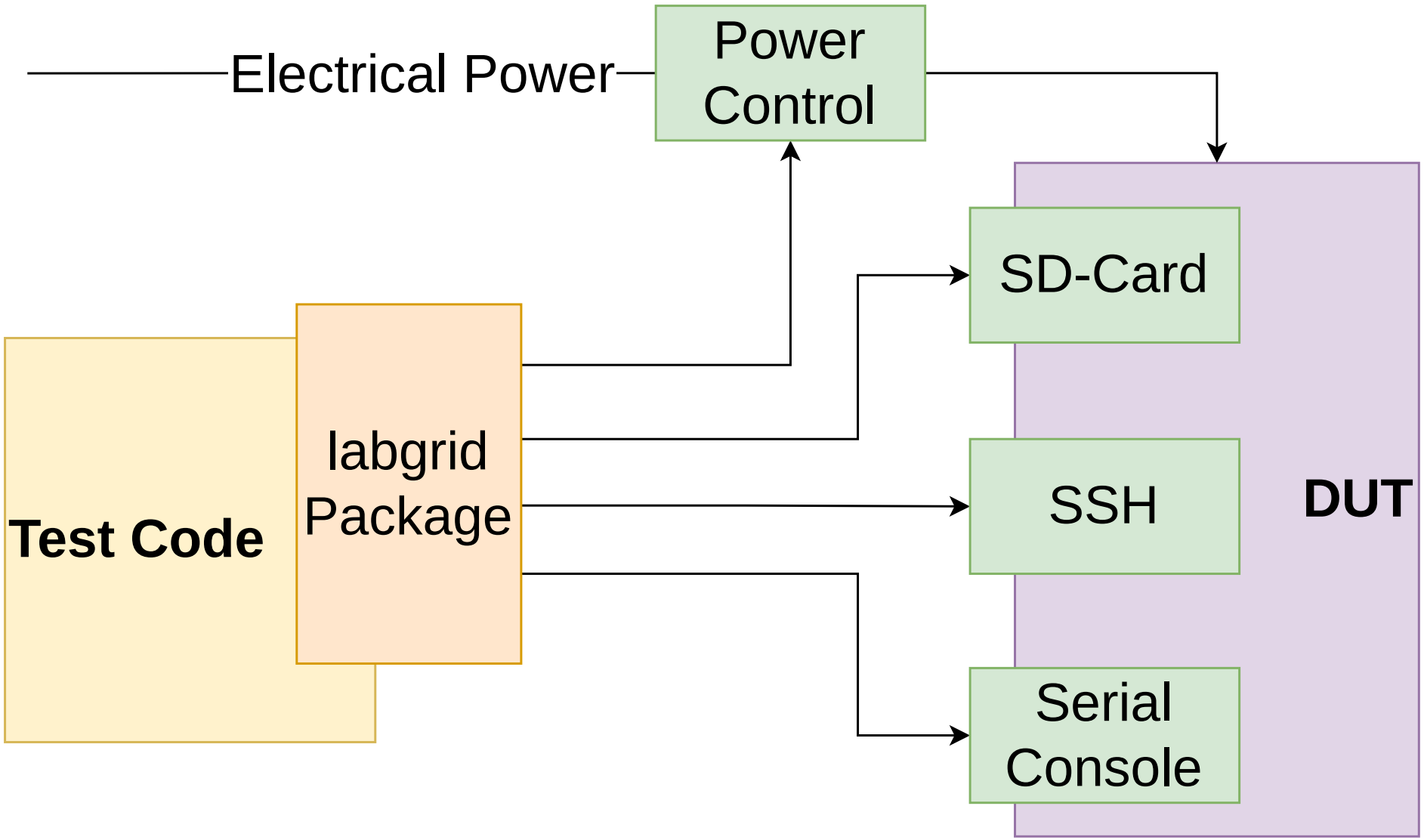

Figure 7: Using the Labgrid Library

honeytree Labs

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)

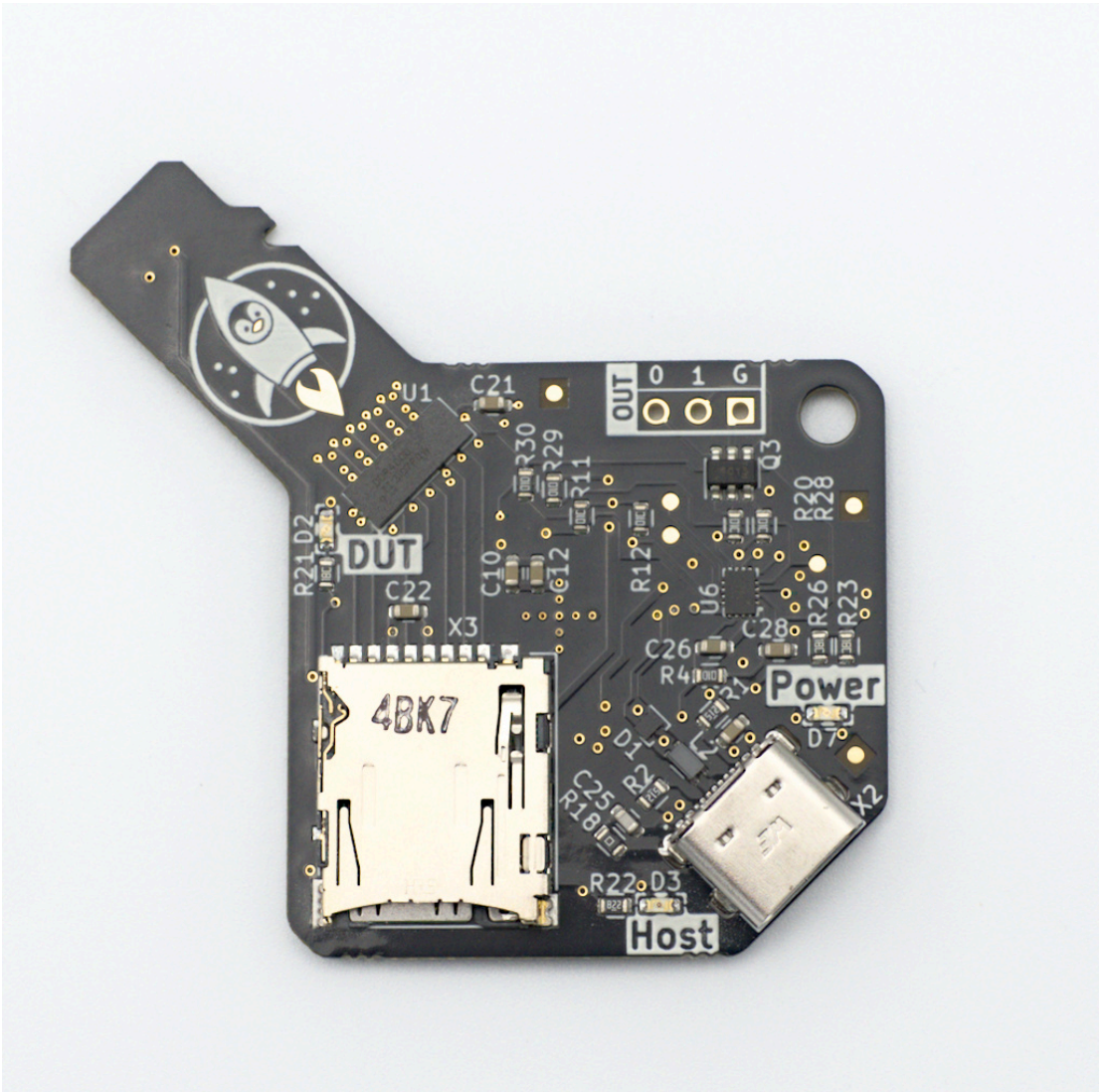

Figure 8: USB-SD-Mux FAST

honeytree Labs

Practical Example

Task: Test the OpenVPN client of the OpenWrt Distribution

honeytree Labs

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

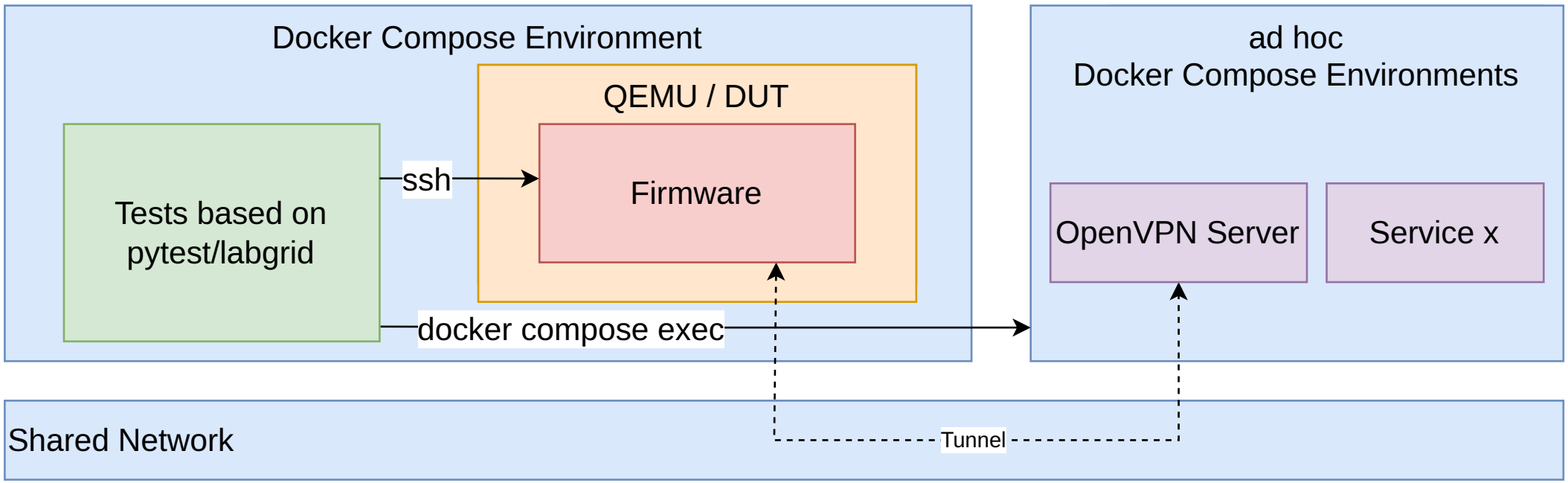

Figure 9: Labgrid QEMU Sample

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)

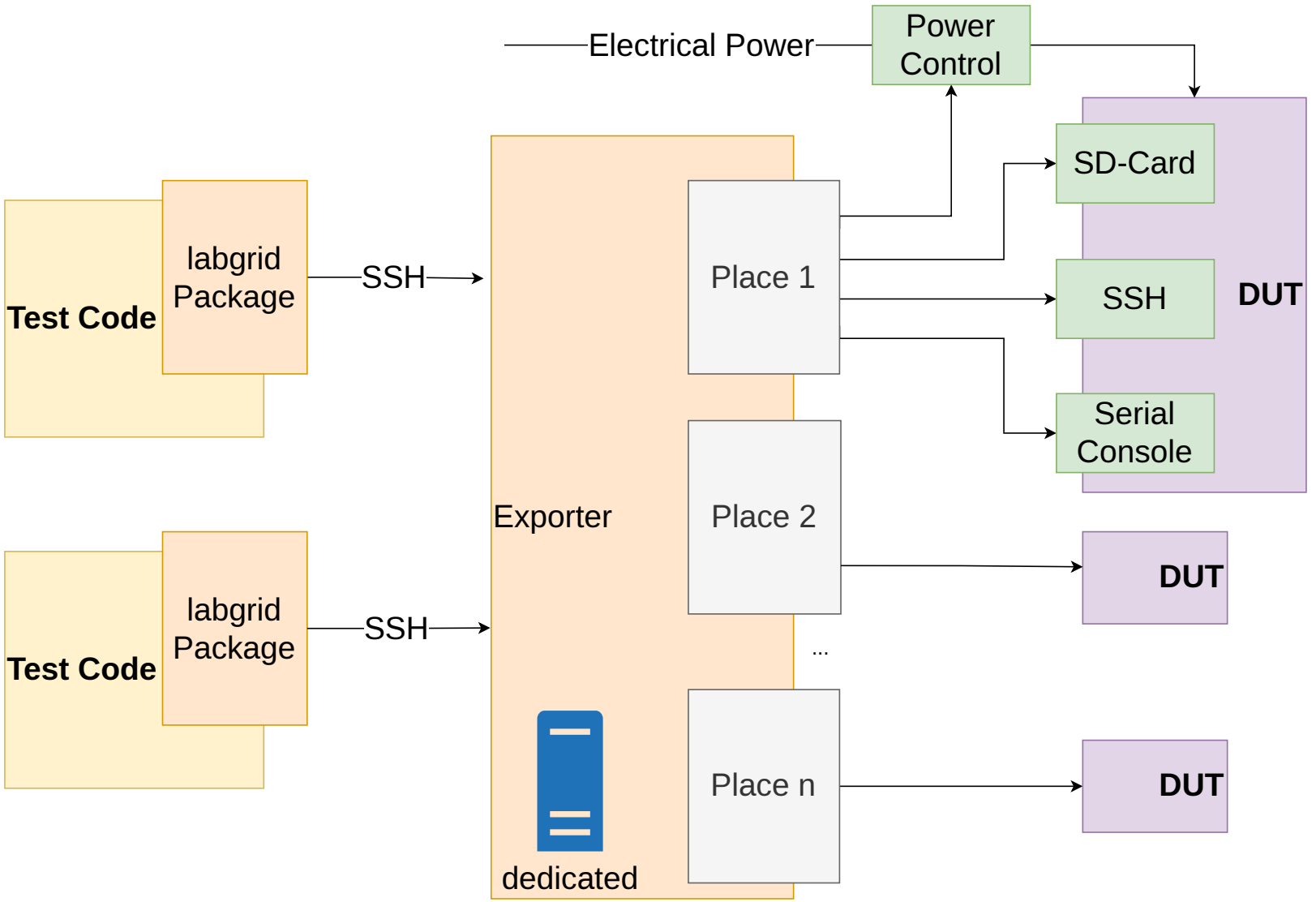

Figure 10: Distributed Labgrid Architecture

honeytree Labs

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/

Our Services

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

honeytree
Labs

Backup Slides

pytest: Fixtures (I)

```python
import pytest
import smtplib


def test_ehlo() -> None:
    smtp_connection = smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
    response, msg = smtp_connection.ehlo()
    assert response == 250
    assert b"smtp.gmail.com" in msg


def test_noop() -> None:
    smtp_connection = smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
    response, msg = smtp_connection.noop()
    assert response == 250
```

pytest: Fixtures (II)

```python
import pytest
import smtplib


@pytest.fixture(scope="module")
def smtp_connection() -> smtplib.SMTP:
    return smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
```

Figure 11: Fixture Definition

```python
def test_ehlo(smtp_connection: smtplib.SMTP) -> None:
    response, msg = smtp_connection.ehlo()
    assert response == 250
    assert b"smtp.gmail.com" in msg


def test_noop(smtp_connection: smtplib.SMTP) -> None:
    response, msg = smtp_connection.noop()
    assert response == 250
```

Figure 12: Fixture Verwendung