About the joy and tears of testing Embedded Devices

Chris Fiege <cfi@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
imeline SmithChart
imeline SmithChart
Agenda

1) Why lab automation?
2) Labgrid overview
3) Demos!
4) Lessons learned
Controlling an Embedded Linux Device?
Motivation for Lab Automation?

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

© https://www.needpix.com/photo/download/697794/
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 command line interface

pytest scripting interface

Serial, Power, GPIOs, Eth, USB:

Fastboot, Bootstrap, Mass-Storage, many more
Distributed Architecture: labgrid’s View

- HDMI, RGB, DSI
- SD-Card
- USB Host, USB Device
- RS232, UART, RS485, Modbus
- Ethernet, CAN, Wifi, Bluetooth
- Power Supply
- GPIO
- CAN
- USB
- 1-Wire via USB
- Eth
- GPIOs
- USB: Fastboot, Mass-Storage
- many more

- Power Protocol
- ConsoleProtocol
- DigitalOutputProtocol
- BootstrapProtocol

Exporter

Coordinators

labgrid command line interface

test

Resources
Distributed Architecture: labgrid’s View

- PowerProtocol
- ConsoleProtocol
- DigitalOutputProtocol
- BootstrapProtocol

Exporter

Coordinator

Client

Resources
Distributed Architecture: labgrid’s View

- PowerProtocol
- ConsoleProtocol
- DigitalOutputProtocol
- BootstrapProtocol
- HDMI, RGB, DSI
- SD-Card
- USB Host, USB Device
- RS232, UART, RS485, Modbus
- Ethernet, CAN, Wifi, Bluetooth

- Place
- Client
  - Exporter
  - Coordinator
- Power Protocol
  - ConsoleProtocol
  - DigitalOutputProtocol
  - BootstrapProtocol
- Place

- Client
  - Exporter
- Coordinator
- PowerProtocol
  - ConsoleProtocol
  - DigitalOutputProtocol
  - BootstrapProtocol

- Labgrid
- Command Line Interface
- pytest

- Serial
- PDU
- GPIOs
- Eth
- USB:
  - Fastboot
  - Mass-Storage
- many more

- PowerProtocol
- ConsoleProtocol
- DigitalOutputProtocol
- BootstrapProtocol
Distributed Architecture: labgrid’s View

- PowerProtocol
- ConsoleProtocol
- DigitalOutputProtocol
- BootstrapProtocol

**DUT**

- Power Protocol
- ConsoleProtocol
- DigitalOutputProtocol
- BootstrapProtocol

**Target**

**Exporter**

- Client

**Coordinator**

- Exporter

**Target**
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

- ⊕ Handling of any special or edge cases for a DUT
  - labgrid allows custom code in test suites
- ⊖ Added complexity for full hardware control
Lessons Learned

Strategies are also useable for interactive and scripting
Lessons Learned

Strategies are also usable 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

Lessons Learned

Only Locking + Reservation and no Scheduler
Lessons Learned

Only Locking + Reservation and no Scheduler

- ☹ CI already has a scheduler, no need to re-invent the wheel
Lessons Learned

Only Locking + Reservation and no Scheduler

- ⊕ CI already has a scheduler, no need to re-invent the wheel
- ⊖ Relies on developers to unlock their DUTs for CI/CT to run
Lessons Learned

Dynamic Resources
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

- ⊕ DUTs can be shared and accessed from everywhere
- ⊕ Noisy or large DUTs can be outside of your office
- ⊖ System is complex: more moving parts involved
- ⊖ 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