Board Farms for Everyone!

Making hardware debugging easier and sharing boards across the globe

Christopher Obbard
Senior Engineer, Collabora Ltd.
chris.obbard@collabora.com
Disclaimer

- Opinions are my own
- Beginner-level presentation
- Aimed at someone who wants to setup personal board farm
- Not a detailed LabGrid presentation
- Suggestions/improvements to my farm welcome
Agenda

• Background & goals
• Review of existing board farm solutions
• How I setup a cheap, functional home lab
  – Software stack
  – Hardware choices
• Labgrid demo
• Future ideas / pain-points / TODO
• Questions
Who am I?

- Hi, I’m Chris!
- Hardware development background => Embedded Linux, IOT
- SE at Collabora in “core” team
- Help customers with system integration
- Custom Embedded platforms
- Debian/Apertis integration
Background / motivation

- Always been interested in board farms
- Learn about Labgrid and Ansible

- Remote working: hard to get a board to “try something out”
- Can debug away from the board
- Flashing SD cards!
- Share limited hardware, short setup time
- Saves planet a little by not shipping boards around
- Automated testing on the same setup
Goal: Setup home lab (May 2022)

- Start small and simple!
- For ~10 devices, ability to scale up later
- Add/remove/reconfigure devices quickly
- Low-power, quiet (WAF)
- No cutting wires, soldering, electrocution risk
- Share boards with colleagues
- Work with non-Linux boards
- Interactive & automated tests from GitLab CI
- Help others create their own lab
Existing solution: LAVA

- Linaro Automated Validation Architecture
- Web interface & backend
- Schedules specific tests, reports results & keeps logs
- Tests are written in YAML, defines execution steps
- Requires database to keep state & logs
Existing solution: LAVA

- KernelCI uses LAVA
- High buy-in; heavy maintenance investment
- MesaCI, other projects also
- Collabora has lab with ~45 different devices; total of ~250
- & always growing
- Anyone at Collabora can submit test jobs to run on real HW
- See Laura Nao’s presentation yesterday
Cambridge LAVA farm
LAVA = not suitable

- Suited to testing specific parts of a system; kernel or graphics userspace
- Suited to testing Linux systems
- Doesn’t support flashing full disk images OOTB
- Debug feedback loop is too long
- Difficult to connect to board interactively
  - Lavabo (inactive)
- Too complicated to setup & maintain for a home lab
Other exiting solutions

- See [https://elinux.org/Board_Farm](https://elinux.org/Board_Farm)
- Long-and-short: not suitable for interactive usage
What do we need for interactive use?

- Control power
- Flash firmware
- Boot into software-under-test
  - TFTP, NFS
  - eMMC, SD card
- Shell
- Abstract nitty-gritty detail away from the user
- Know who’s using what hardware
Enter... Labgrid

- Developed by Pengutronix (LGPL)
- CLI (interactive) and Python library (automated) usage
- Usable for daily interactive development/debugging
- Abstracts the hardware interaction into simple functional topics, i.e. turn power on, flash image
- Three bits of software:
  - (one) Controller handles overall system state
  - (many) Exporters control specific hardware
  - (many) Clients on PC / CI runner connect to the Controller
Controller/Exporter

- May reside on same machine
- A Controller may have multiple Exporters connect to it
- An Exporter has multiple Places
- A Place is a collection of multiple Resources which are connected to a specific board
- A Resource is just hardware connected to an exporter
  - e.g. NetworkPowerPort is a PowerPort: switch on/off/state
  - e.g. USBSerialPort is a SerialPort: get to console
Labgrid Configuration

- YAML
- exporter.yaml defines the devices on an exporter
- Use labgrid-suggest to bring up new devices
Labgrid Configuration

- places.yaml coordinator configuration
- Generated by
  - `labgrid-client -p foo create/add-match`
  - `labgrid-client -p foo set-tags`

```yaml
places:
  rockchip_rk3399-nanopi-m4-0001:
    matches:
      - "*/rockchip_rk3399-nanopi-m4-0001/*"
    tags:
      board: rk3399-nanopi-m4
      soc: rk3399
```
Using Labgrid Client

(note: it may be preferred to install in a virtualenv; see installation instructions for more details)

$ sudo apt install labgrid

$ export LG_CROSSBAR=ws://labgrid-exporter-0001:20408/ws

$ labgrid-client places
rockchip_rk3328-rock-pi-e-0001
rockchip_rk3328-rock-pi-e-0002
rockchip_rk3399-nanopi-m4-0001
rockchip_rk3399-roc-pc-0001
Using Labgrid Client

$ labgrid-client resources
labgrid-exporter-0001/rockchip_rk3328-rock-pi-e-0001/TasmotaPowerPort
labgrid-exporter-0001/rockchip_rk3328-rock-pi-e-0001/NetworkSerialPort
labgrid-exporter-0001/rockchip_rk3328-rock-pi-e-0002/TasmotaPowerPort
labgrid-exporter-0001/rockchip_rk3328-rock-pi-e-0002/NetworkSerialPort
labgrid-exporter-0001/rockchip_rk3399-nanopi-m4-0001/TasmotaPowerPort
labgrid-exporter-0001/rockchip_rk3399-nanopi-m4-0001/NetworkSerialPort
labgrid-exporter-0001/rockchip_rk3399-roc-pc-0001/TasmotaPowerPort
labgrid-exporter-0001/rockchip_rk3399-roc-pc-0001/NetworkSerialPort
Using Labgrid Client: locking

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 acquire
acquired place rockchip_rk3328-rock-pi-e-0001

# The device is yours to debug with.

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 release
released place rockchip_rk3328-rock-pi-e-0001
Using Labgrid Client: locking

```sh
$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 acquire
labgrid-client: error: place rockchip_rk3328-rock-pi-e-0001 is already acquired by nemesis/chris

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 unlock
labgrid-client: error: place rockchip_rk3328-rock-pi-e-0001 is acquired by a different user (nemesis/chris), use --kick if you are sure

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 unlock --kick
warning: kicking user (nemesis/chris)
released place rockchip_rk3328-rock-pi-e-0001

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 acquire
acquired place rockchip_rk3328-rock-pi-e-0001
```
Power

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 power get
power for place rockchip_rk3328-rock-pi-e-0001 is off

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 power on

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 power off

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 power cycle
Console

$ labgrid-client --place rockchip_rk3328-rock-pi-e-0001 console
cnecting to NetworkSerialPort
connected to ::1 (port 38063)
Escape character: Ctrl-\nType the escape character to get to the prompt.

DDR3, 333MHz
BW=32 Col=10 Bk=8 CS0 Row=14 CS=1 Die BW=16 Size=512MB
Trying to boot from BOOTROM
Returning to boot ROM...
<etc>
Generic Labgrid drivers

- Console: SSH, Serial port, USB serial, Network connection
- Power: pdudaemon, PoE port, Tasmota, USB Relays, USB socket, Sigrok, GPIO...
- Flashing: IMX Loader, RK loader, Fastboot
- Digital outputs: GPIO, Relays
- SO many more, USB flash drives, SD muxes, HDMI/webcam capture, audio....

- Implemented as simple python classes
Flashing...

- Flash SD card: SD Mux adapter is expensive
- Can flash over JTAG using OpenOCD
- Can upload images using Fastboot
- Flash Rockchip bootloader to eMMC over USB

- Currently my boards TFTP boot kernel/nfsroot
  - Manually SCP files after build to /srv/nfs
  - Boards have image with bootloader on SD card (set to just TFTP boot)
- Raspberry Pi 4
- USB hub for UART
- Devices on own VLAN
- Power delivery through controlled sockets
Raspberry Pi software configuration

- Install Raspbian & change hostname
- **Ansible playbooks**
- One Ansible playbook sets everything up!
  - Includes nfs server, MQTT server
  - Everything in containers
  - Edit Labgrid configuration to suit your project
Power: Relay board

- Robot electronics
- Requires cables to be cut
- Messy wiring
- Relays are supposed to switch AC
- DC reduces lifespan / welded contacts

dS3484 - 4 x 16A ethernet relay
- £77.96
- Excl. Tax: £64.97
- As low as £58.57

USB-RLY08C - 8 channel USB relay
- £47.99
- Excl. Tax: £39.99
- As low as £36.06
Sonoff WiFi sockets

- Plug any adapter in
- No soldering needed
- Turn on/off manually
- Fairly cheap
- Seem to be reliable
- ESPHome firmware
- Firmware update over WiFi
Side note: Home Assistant integration
Console

• Just using USB→UART adapters
  - Clone FTDI adapters
  - Silicon Labs adapter with additional GPIO pins
• Note: Buggy with USB hub
Remote sharing

- Exporters need to reach the coordinator locally
- Labgrid client can proxy connections to farm via simple ssh tunnel
  - `labgrid-client -P <PROXY>`
CI

- GitLab runner on same machine as exporter
- Shell script mirrors interactive mode:
  - lock the device (2hr timeout)
  - copies the NFS file to exporter
  - power on the device
  - runs test script/commands on board & compares output
  - powers off the device
- Write your own test with pytest; examples on GitHub
Future Labgrid improvements

- Build Docker containers for arm64 (WIP)
- ESPHome API for power / GPIOs (WIP)
- USB Serial Port GPIO driver
- Flash Allwinner bootloader with sunxi-fel
- Ability to rsync files to TFTP/NFS server from labgrid-client
Future Lab improvements

- Documentation!
- More boards!
- Healthchecks
- Ability to flash boards over USB
- More PDU slots / generic power slots
  - Create an open-hardware extendable PDU?
Where can I learn more about Labgrid?

- Excellent setup video by Pengutronix
- Documentation: https://labgrid.readthedocs.io
- Source code: https://github.com/labgrid-project/labgrid
Special thanks

• Da Xue – Libre Computer
  – Donation of ROC-RK3399-PC

• Tom Cubie – Radxa
  – Donation of 2x Rock Pi E

• Julien Snell – Cocom Consumer Electronics
  – Donation of NanoPi M4

• Jan Lübbe – Pengutronix
  – Slide review
Thank you! Questions?

We are hiring

col.la/careers