



Dasharo Open Source Firmware Validation



Agenda

- Introduction to Dasharo Open Source Firmware Validation (OSFV)
- Exploring OSFV v0.2
- Recent improvements
- Getting started with OSFV
- Upcoming Features: OSFV v0.3 and Beyond
- Q&A

History

The screenshot shows the GitHub repository page for `open-source-firmware-validation`. The repository is public and has 16 branches and 1 tag. The current branch is `main`. The repository description is "OSFV infrastructure with automated tests and scripts for managing test results". The repository has 3 stars, 6 watchers, and 0 forks. The repository was created 5 hours ago and has 17 commits. The repository contains several folders: `.github/workflows`, `dasharo-compatibility`, `dasharo-performance`, `dasharo-security`, `dasharo-stability`, and `keys-and-keywords`. The repository is licensed under MIT.

Folder	Commit Message	Time
<code>.github/workflows</code>	<code>.github/workflows/test.yml: test workflow</code>	last week
<code>dasharo-compatibility</code>	<code>tree-wide: use less flags for reboot/suspend iterations</code>	last week
<code>dasharo-performance</code>	<code>pre-commit: fine-tune RF rules, more fixes</code>	last week
<code>dasharo-security</code>	<code>dasharo-compatibility/secure-boot: add more cases</code>	5 hours ago
<code>dasharo-stability</code>	<code>tree-wide: use less flags for reboot/suspend iterations</code>	last week
<code>keys-and-keywords</code>	<code>pre-commit: fine-tune RF rules, more fixes</code>	last week

- We've been using OSFV at least since 2018, when we've started validating PC Engines coreboot releases on a monthly basis
- Using those scripts we've executed over **50k** tests publicly releasing **150+** binaries based on open-source firmware
- Initially, it was closed source because of assumption that validation provides majority of the value in open-source firmware development

<https://github.com/Dasharo/open-source-firmware-validation>

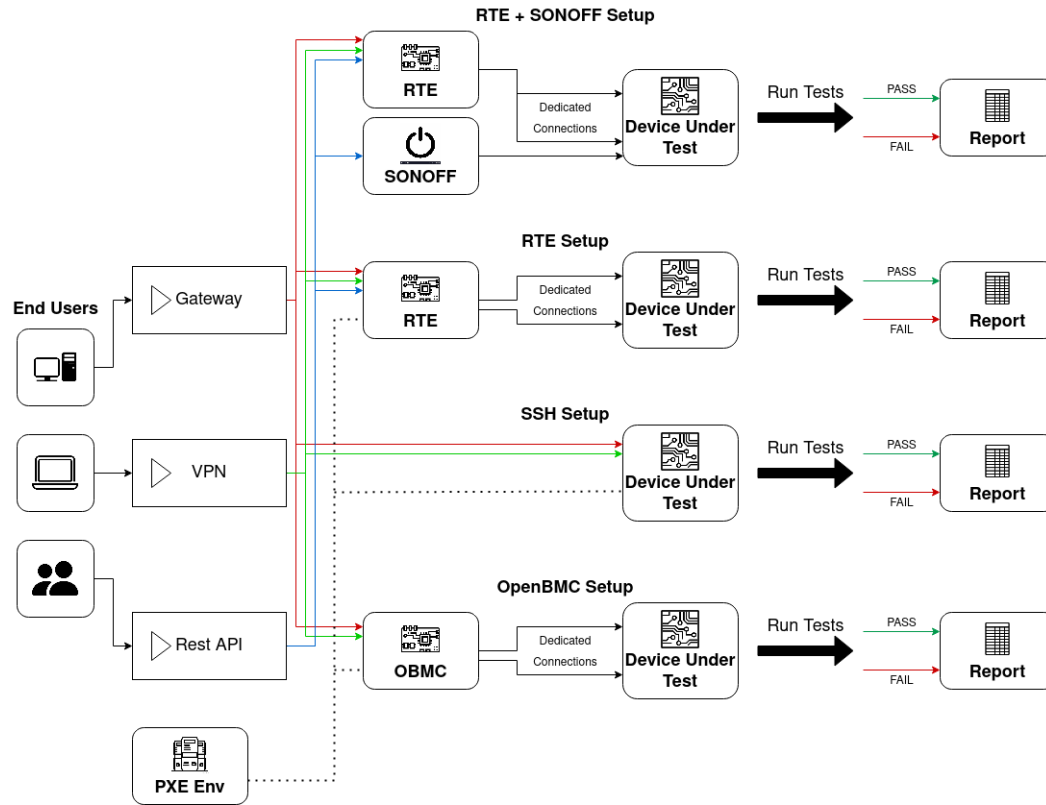
- Since then, these scripts came through many iterations, supporting more different products
- At some point we have decided to open-source what we have and start maintaining and improving it as an open-source product
- Dasharo Open Source Firmware Validation purpose is the validation of open-source firmware (mainly Dasharo)
- Scripts written in:
 - mostly Robot Framework (keywords, test suites)
 - some Python (for some keywords, sometimes more suitable than RF)
 - shell scripts (mostly some wrappers for test execution)

Robot Framework

- Robot Framework is a generic open source automation framework
- It can be used for test automation and Robotic Process Automation (RPA)
- Used widely for web apps testing (with Selenium), but not only
- Used by OpenBMC (firmware, embedded Linux) for test automation
 - <https://github.com/openbmc/openbmc-test-automation>
- Active community, quality documentation
 - <https://robotframework.org/#community>
 - <https://docs.robotframework.org/docs>
- Robot Framework Conference
 - <https://robocon.io/>



Regression Testing Architecture



OSFV Supported Platforms

Supported platforms [↗](#)

Manufacturer	Platform	Firmware	\$CONFIG
NovaCustom	NV41MZ	Dasharo	<code>novacustom-nv41mz</code>
NovaCustom	NV41MB	Dasharo	<code>novacustom-nv41mb</code>
NovaCustom	NS50MU	Dasharo	<code>novacustom-ns50mu</code>
NovaCustom	NS70MU	Dasharo	<code>novacustom-ns70mu</code>
NovaCustom	NV41PZ	Dasharo	<code>novacustom-nv41pz</code>
NovaCustom	NS50PU	Dasharo	<code>novacustom-ns50pu</code>
NovaCustom	NS70PU	Dasharo	<code>novacustom-ns70pu</code>
MSI	PRO Z690 A WIFI DDR4	Dasharo	<code>msi-pro-z690-a-wifi-ddr4</code>
MSI	PRO Z690 A DDR5	Dasharo	<code>msi-pro-z690-a-ddr5</code>
Protectli	VP2410	Dasharo	<code>protectli-vp2410</code>
Protectli	VP2420	Dasharo	<code>protectli-vp2420</code>
Protectli	VP4630	Dasharo	<code>protectli-vp4630</code>
Protectli	VP4650	Dasharo	<code>protectli-vp4650</code>
Protectli	VP4670	Dasharo	<code>protectli-vp4670</code>
Raptor-CS	TalosII	Dasharo	<code>raptor-cs_talos2</code>

Warning [↗](#)

!!! **WARNING** !!! This repository is in the process of migration and multiple major reworks. If you do not know what you are doing, consider not using it until at least `v0.5.0` is released. When this is scheduled, link to such a milestone will appear here. !!! **WARNING** !!!

- v0.2 was a silent release (no announcements)
- It started very active migration from our private repositories to the public
- The goal is to start adoption among Dasharo and open-source firmware developers

Recent improvements

- Reduced `codebase size` from ~4000 lines to ~2000 lines
 - we are splitting that into more manageable modules under
- Added `pre-commit hooks` to enforce rules on commit and code style
 - [robocop](#)
 - [robotidy](#)
 - [black](#)
 - [conform](#)
- Applied these rules tree-wide



Recent improvements

- Added support for running selected tests in QEMU (using Dasharo OvmfPkg release for QEMU)
 - https://docs.dasharo.com/variants/qemu_q35/releases/
 - <https://github.com/Dasharo/edk2/releases>

```
Standard PC (Q35 + ICH9, 2009)                2.00 GHz
pc-q35-7.2                                    1024 MB RAM
0.0.0

Select Language          <Standard English>
                          This is the option
                          one adjusts to change
                          the language for the
                          current system

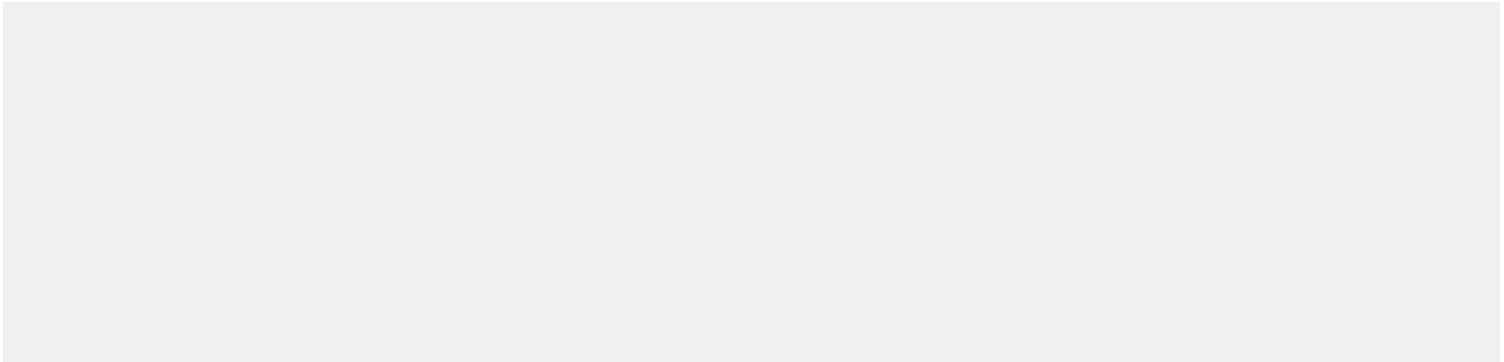
▶ User Password Management
▶ Device Manager
▶ Dasharo System Features
▶ One Time Boot
▶ Boot Maintenance Manager

Continue
Reset

↑↓=Move Highlight      <Enter>=Select Entry
LCtrl+LAlt+F12=Save screenshot
```

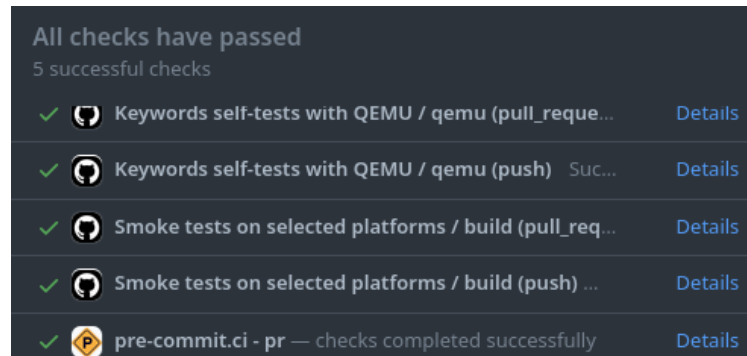
Recent improvements

- Added tests under `tests` directory
 - a form of unit tests, testing if basic keywords (such as moving in the Dasharo menus) still function as expected
 -



Recent improvements

- Reworked keywords for parsing and moving around in the Dasharo menus (see: [#100](#))
 - better code readability reduced duplicate keywords
 - improved reliability (entering to the desired menus, even if something changes in menus, such as new options gets added)
- Added CI checks
 - pre-commit checks
 - keywords [#100](#) in QEMU
 - a basic test on two hardware units: [#100](#), [#100](#)



OSFV v0.3 and beyond

- **variables** - use Python/YAML, not robot syntax, currently used variables system for keeping configuration is not robust to be used at scale (too error prone), we have to switch to something manageable
- **platform-configs** - get rid of duplication, and unused data, simplify way of adding new platforms,
- **improve tests organization** - separate test for different OS into different suites, look for inspiration in other projects about test organization,
- **prepare the OS for running test suite via some dedicated tools (e.g. Ansible, SaltStack)**, rather than implementing keywords for that from scratch - the problem is that writing robot keywords is expensive and RF is not dedicated tool for system configuration

OSFV v0.3 and beyond

- **reduce the number of unnecessary power events** - too many power cycles drain energy and make tests slow,
- improve overall code quality by enabling back more robocop checks we cannot pass right now,
- Enforce documentation sections for all keywords and tests
- Automatically generate and publish documentation for keywords and tests
 - keywords - useful for developers
 - tests - useful for following the test scenario and reproducing it manually
 - the goal is to document tests via automated tests, not via dedicated test manuals
 - it is expensive to maintain this separately
 - it is difficult to keep it in sync with automated cases

Getting started with OSFV

- What do I need to learn first?
- How do I run existing test?
- How do I write a new test?
- How do I add support for a new platform?

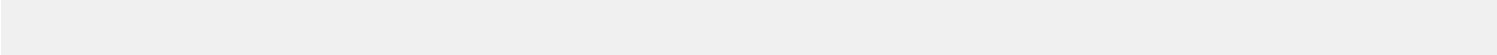
What do I need to learn first?

- Some basic RF knowledge
 - go through basics in [User Guide](#):
 - go through basic [RF libraries](#):
 - [RF libraries](#), [RF libraries](#), [RF libraries](#)
 - add them to your bookmarks, you will need them often
 - check out [SSHLibrary](#).
- Some basic Python knowledge
 - there are plenty of learning materials, pick your favourite one

How do I run existing test?

- Consult README for:
 - [supported platforms](#)
 - [getting started](#)
 - [running single tests](#)
- Look through existing tests in:
 -
 -
 -
 -

How do I run existing test?

- Check if selected test is supported by the given platform
- In :


- 

- Example on hardware:

How do I write a new test?

- Refer to the existing tests
 - are good examples
 - other commonly used tests
 -
 -
 -
 -
 -
 -
 - some tests may currently not work or be of a lower quality

How do I add support for a new platform?

- Pick a similar board from _____ and adjust it
- Power control
 - [RTE, Sonoff WiFi Smart Plug](#), or both (or something else, which is not supported yet)
- Flashing
 - preferably external flashing is supported - like SOIC clip connected to RTE all the time
- Serial connection
 - [ser2net service](#) to expose serial via telnet
- Hardware setup may be complex
 - https://docs.dasharo.com/variants/asus_kgpe_d16/setup/

Contributing to OSFV repository

- Many parties can contribute there
- By contributing you get the benefits such as:
 - access to test developed by 3mdeb or other parties
 - stable environment tested in more scenarios
- 3mdeb maintains repository to make sure changes from external parties does not break others, and can be shared between them
 - access to all supported platforms
 - experience
- Standard GH issues and PR flow for contributors
- Join Dasharo Matrix Space <https://matrix.to/#/#dasharo:matrix.org>
- Join

Dasharo	OSFV	Matrix	room:
https://matrix.to/#/#osfv:matrix.3mdeb.com			



Q&A