Fuego Test System
Projects, Industry Initiatives, and Vision

June 2018

Tim Bird

Fuego Maintainer

Sony Electronics

Outline

Projects

Industry Initiatives

Vision

Outline

- Features in progress
  - 1.3 stuff that got missed
  - underutilized features:
    - dependencies, dynamic variables, charting, criteria sharing
  - dynamic documentation
- Industry initiatives
  - Board automation standards
  - Definition of Automated Testing stack
  - Automated Testing Summit
  - Kernelci Linux Foundation project
- Vision
  - easy customization
  - test server

Missing from 1.3...

- Things that slipped from the 1.3 release:
  - Documentation conversion
  - LTS Provisioning support
  - Pre-built docker

Documentation conversion

- Conversion of docs to reStructuredText
  - Replace PDF and wiki docs with rst
  - Move all docs under source repository
  - Use sphynx to create multiple formats
  - Publish on readthedocs.io
- Made some progress
  - Have sphynx templates in place
- Got stuck on markup conversion
  - Considered automation, but hit some hurdles
- See http://fuegotest.org/wiki/rst_docs

LTS Provisioning support

- Provisioning
  - Ability to provision board with new system software (particularly the kernel)
  - Fuego historically has left this as an exercise for the user
- Did some work on this in my lab
  - usb keyboard automation
    - teensy-usb – host-controlled keyboard for target
  - LTS download and build
  - Ubuntu kernel replacement
  - Haven't generalized the feature
    - Some support was put into ttc

Pre-built docker image

- Ability to use Fuego without building the docker image
  - Create a pre-built Fuego docker image, and host it at docker.io
  - e.g. “docker run fuego”
- Requires automatic container customization
  - Network proxy
  - User and group
  - Volume mount customization
- Includes refactoring the Fuego directory layout
  - Turned out to be too intrusive for 1.3 release

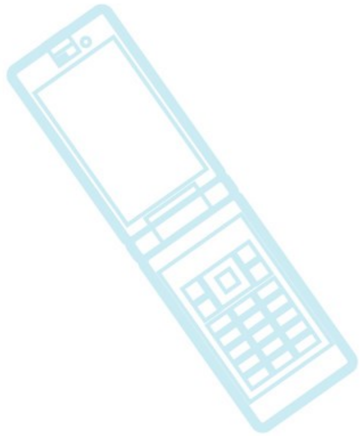
Underutilized features

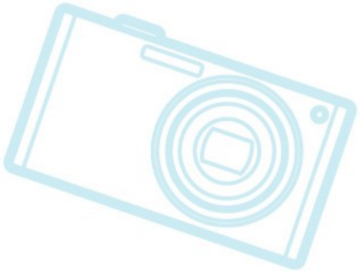
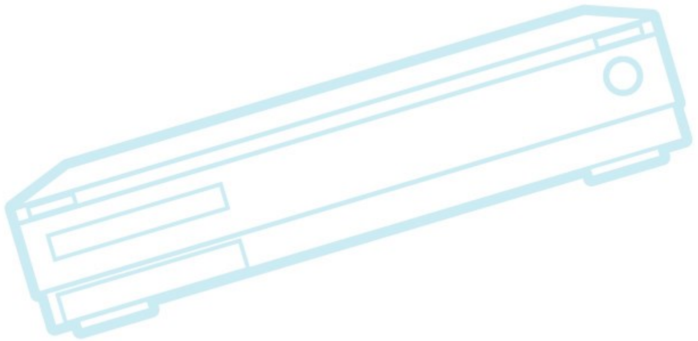
- Overlay system
  - Ability to override any ‘ov_’ function
- Dynamic board variables
  - Intended for automatic test customization
  - Dependency information cache
  - Automatic installation
- Customized (per-board) pass-criteria
- Specs

Dynamic Documentation

- Provide documentation on a per-suite, per-testset and per-testcase basis
- Allow users to share well-structured information about a test
  - test outline, expected results, notes
  - links to resources
- Integrate dynamic report (generated with ‘ftc gen-report’) into text
- Have .rst template system
  - Not generating anything from rst yet.

Outline



Projects

Industry Initiatives

Vision

Test System problems

- No “lego blocks” for test system infrastructure
- Current systems are monolithic
  - e.g. Hard for Fuego to use LAVA as board control software
  - Have mismatches in models, artifacts
- Lots of islands of work
- Nobody handles off-DUT hardware orchestration
  - Maybe LAVA, but it’s not generalized
    - (e.g. LAVA multi-node tests)

Automated Test Standards

- Would be good to define:
  - objects, methods, interfaces, protocols
- Want to mix and match test stack layers, and allow separate implementations to compete
  - board control
  - test orchestration
  - results parsing
  - results aggregation
  - analysis, etc.
- Reuse features from other domains
  - e.g. log results visualization
  - e.g. libvirt for hardware board control

Previous discussions

- Presentation at Linaro Connect
  - See http://fuegotest.org/ffiles/Test-Standards-LC-2017.pdf
- Lots of meetings at ELCE 2017 on this
  - Pengutronix introduced labgrid
  - Linutronix demonstrated r4d and libvirt
  - BOF resulted in some collaboration:
    - See https://elinux.org/Board_Farm
    - Mailing list for discussion:
      - https://lists.yoctoproject.org/listinfo/automated-testing
- Please join this discussion

Automated Testing Summit

- October 25, Edinburgh Scotland
  - See http://elinux.org/Automated_Testing_Summit
- Sponsored by Linux Foundation Core Embedded Linux Project
- Attempt to assemble wide variety of Linux test stakeholders and practitioners
- Likely a by-invitation meeting
- Add name to wiki page or e-mail me if you are interested in attending

KernelCI Linux Foundation Project

- Kevin Hillman has proposal for making a Linux Project to support KernelCI
- Project is being done by individuals in spare time (is underfunded, can't expand)

Outline

Projects

Industry Initiatives

Vision

Fuego

Vision – super high level

Do for testing
what open source
has done for coding

- Significant parts of the test process are unshared, ad hoc, private, etc.
  - For no good reason – most QA doesn't need to be proprietary
  - There are OSS frameworks and test programs but parts are missing to create a open testing community.
- Fuego Goal:
  - *Promote the sharing of tests, test methods, and results, the way code is shared now*
    - Make it easy to create, share and discover tests
    - Make test results easy to share and evaluate

Core principles

- Useful
  - Actually find bugs or prevent regressions
- Scales
  - Allow sharing
  - Usable by wide audience
    - Minimal requirements
    - Customizable
    - Easy to use
  - Modular
- Applicable to embedded

Sharing, Generalization, and Customization

- Sharing is key to reduce effort
- Tests have to be generalized so others can run them in different environments
- Customization is important to be able to leverage what is shared
- Fuego has good artifacts that can be shared
  - However, people are only sharing tests so far
- Fuego has good customization in 3 of 4 areas:
  - test applicability (dependencies)
  - test instantiation (specs)
  - test results analysis (pass-criteria)
  - expected values (???)

Easy test customization

- Customizable system state check
  - Ability to save a snapshot of system status or behavior
  - Example:
    - “I want the system to still have X”
    - “I want the system to continue to be able to do Y”
- Make it very easy to capture state or behavior as an “expected value”
- Reduces maintenance for tests

Work in progress

- clitest
- Functional.fuego_board_status
- Functional.fuego_compare_reports
  - meta-test – tests results of other tests
  - ‘save_baseline’ spec
    - Run to establish baseline report
  - ‘default’ spec
    - Run to compare current report with baseline
- seddiff
  - compare with regex masking

Demo of tbwiki regression test

- Update expected value very easily
  - Just a few mouse clicks

Test Server

- Fuego centralized test server
  - Test artifact sharing hub
    - tests, specs, criteria, boards, results, run-requests
  - Test store
  - Request dispatcher
- Use cases:
  - Share ad-hoc test (test package)
  - Request test on someone else’s board
    - Allow developer to see results from a wide variety of boards
    - Test on hardware that a developer doesn’t have locally
  - Mine data for patterns
  - Use customizations for your testing

Fuego

Fuego Features

- Pre-Built docker image
  - Eliminate long Fuego install step
- Test program binary cache
  - Remove need for SDK in order to test
- Focus on pass-criteria customization and sharing
  - For testplan_lts tests, to remove false positives

Provisioning and scaling the testing effort

- Automated provisioning
  - Requires hardware control for 100% reliability
  - Less than 1% of users will use hardware to automate their kernel installs
  - Want to support semi-automated provisioning
- Trying hard in Fuego to avoid requiring hardware board control
- "Semi-automated" means:
  - Try software board control, and fall back to user intervention