RIOT Summit 2020 - HiL Testing Breakout Session

Goal

The purpose of this breakout sessions is to communicate somewhat fragmented testing efforts.
We will share some knowlage/experiences and ask some questions to the community.
Notes will be taken and later compiled and email, following that some tracking issues or RDMs will be made.

Topics

- Problems with testing
- Emulation
  - Renode
  - QEMU
  - native
  - Integration with CI
  - Supported emulation features in makefiles
  - Connecting emulated boards
- Test Repos
  - RIOT
  - RobotFW-Tests
  - Release-Specs
- CI
  - Murdock
  - jenkins hil
  - paper-ci
  - nightlies
- Test interfaces
  - riotctrl
  - riot_pal
  - test interface for firmware
  - ci for test interfaces
- Display Test Data Information
  - xunit
- Tools to help testing
  - Autopost results in md format
- Future Work
  - Improve/unify robot tests
- RYOT
  - Progress since last summit
  - Current Status

Problems with testing

- People generally don't like writing tests
- Tests are not catching the problems that are needed
- Generally the developer of the feature writes the test, may suffer from tunnel vision but in many cases other do not understand the feature enough to be able to meaningfully contribute to the test

Emulation

RIOT supports emulation for a few boards.
Others use QEMU/renode/fast models/, jumper.io is dead.
Some setup is required and no versioning is in place.

Renode

I really like it, supports multinode, daughter boards, and environment control (temperature).
Easy to have periphs with python but core implementation done in C#.
I tried to get other boards (stm based) working but hacks were needed to deal with the RCC (gets stuck waiting for clock state to change).
In master they have an stm32f4 based RCC module implemented but it isn't released yet.
It would be nice to simulate PHiLIP but needs RCC and DMA to work.
If we can simulate PHiLIP then we can virtually connect and test periphs if RIOT boards and verify the tests with hardware.

- Is renode cycle accurate?

QEMU

It has been around longer, written mostly in C.
At first glace it doesn't seem to target the Cortex Mx series but there are many more implementations if you dig around.
It seems to have less boards but more fully supported.

- Not cycle accurate
- Timer tests dont work

native

Very useful but still has some strange behaviours.
Would it be nice to have native open a serial com to behave like other boards /dev/ttyNATIVE ?
Bootstrapping might be an issue?
You can run native with tcp socket (@miri64 is working on this) should be already in master (https://github.com/RIOT-OS/RIOT/pull/405).
$ examples/gnrc_networking/bin/native/gnrc_networking.elf -h [...] -e, --stderr-pipe redirect stderr to file -E, --stderr-noredirect do not redirect stderr (i.e. leave sterr unchanged despite daemon/socket io) -o, --stdout-pipe redirect stdout to file (/tmp/riot.stdout.PID) when not attached to socket -c <tty>, --uart-tty=<tty> specify TTY device for UART. This argument can be used multiple times (up to UART_NUMOF) [...]
Very sexy.
Timing seems a bit broken on native the last time I looked.
Also the periphs are a limitation, do we want to try to add gpio toggle (maybe working on a pi)?
Could we add periphs that just output values to a terminal or somehow simulate them?

What is happening with 64 bit support?

Integration with CI

I don't think the CI runs any emulation, we would need to add it to the murdock or jenkins image.
We also to identify what would be supported somehow, this could be some combination of info in the makefile and results from tests on hardware.
Maybe it would be good to add the packages to the docker image.
Currently renode master has support for the
rcc module in stm32f4* boards but it is not in the release.
Without it
RIOT hangs at startup of the emulation.
I can imagine that setup of the emulators is not that trivial.

Supported emulation features in makefiles

It doesn't seem like there is a standard way to query which boards support emulation.
The
cc2538dk has
FEATURES_PROVIDED += emulator_renode but it may be nice to standarize this a bit.
KW-> Make a tracking issue to standardize this.

Connecting emulated boards

If board emulation gets to a stable state we can think about multiboard testing and/or connecting daughter boards up.
If so it would be nice to have a way to define recommended connections.
This is done with PHiLIP in the RobotFW-Tests/dist/etc/conf .
Generally it has been a bit of a challenge to locate available pins, it would also be nice to add some tests to ensure pins are match the layout so emulation does not break if a pinout is changed.

Test Repos

There are a number of places where tests for RIOT exist, it would be good to have an overview of them.

RIOT

Total of ~412 tests, actual tests that are automated and can run on boards are around 209 for the nucleo-f103rb and 212 for the
samr21-xpro .
The
unittests also include around 64 tests.
Note that many of the tests are simple compile and flash tests.
Some have test logic in c, some in the python test runner.
Parameterized tests with k-config

RobotFW-Test

Adoption is poor it seems, test coverage is not that large but does some important testing.
Most tests require wiring to PHiLIP making then generally inaccessable to the standard developer who just wants to be confident in merging.
Acts more as a basis for ideas/prototypes (see structured interface, test helpers)
General feedback is not that easy to get into but once understood it is ok.
After long periods away from robot tests it also takes some time to figure things out.

Purpose is mainly peripheral testing.

Release Specs Testing

Now fully automated it with the help of
riotctrl and
pytest .
More focused on network testings though part is running the RIOT tests on boards (these are not run in the release-tests action but the test-on-iotlab action which works similar).
Uses IoT labs and native.
Get run every week in a while on master?
Uses multiple nodes.

CI

RIOT has many forms of Continuous Integration tests, and it seems to be growing fast.

Murdock

Customized tool by Kaspar tuned to help with build tests and distributing jobs.
So far the most efficient option as the workload required to build everything is pretty high (75k jobs?).
Some work has been done recently to update it.
It would be nice to have some additional help (frontend stuff).
Supports some HiL boards with pi-fleet as well.
Not only build test results but also build sizes and (maybe) benchmarks

Jenkins (hil)

CI used in HiL testing with RobotFW-Tests, based on Jenkins.
Pretty complex now, too a while to tune and sort out the config and still has a ways to go.
Fairly reliable now in terms of reporducable results (tests don't seem to toggle as much due to noise).
Still some flashing issues with some boards.
Maybe some interface issues as well (at least with the nrf52dk).
Run on nightlies and per PR in the robot framework tests repo.
Pretty complex setup as each board must be wired in and wiring env must be created.

paper-ci

Idea started by @cladmi to use any CI (in this case Jenkins), to run RIOT tests on boards that are just lying around.
Very simple setup, old computer with boards plugged in, maybe some nice udev rules.
Runs RIOT tests on hardware.
Responible for many improvements to the RIOT testing infrastucture.

Nightlies

- Add fuzzing to nightlies?
- Failures from murdock HiL, should we have warning, instead?

Test interfaces

There are scattered efforts to define interfaces for tests.
It would be good to bring them all together.
The type of interface can be catigorized:

- Firmware input - what the firmware is expecting, eg. r to print ready, s to start tests
- Firmware output - how the firmware prints results, eg. unstructed text so developers can read

and regex needed to parse
- Host interface input - Abstraction that can be used on the host computer, eg. riotctrl or riot_pal
- Host interface output - how the data get reported up, again like riotctrl
- Test artifacts - How results are stored, eg. build_and_test_for_board has file based storage, some xunit stuff, murdock has some test artifact storage, at least for build sizes maybe benchmarks.

riot_pal

Seems a bit silly at this point.
It is basically a poor wrapper of Serial.
Probably better to replace it with riotctrl.
Maybe use riotctrl for a simple setup.

test interface for firmware

There is currently not really a way to properly write test firmwares.
This idea was introduced in a testing rdm but was not completed.
I suppose the ideal way of doing this is with some testing function similar to what is used in robot framework and a way to manage the args better.
This would allow for an abstraction of the test logic and the output/input to the firmware.
With this we can make both console based test firmware (for developers to play around) and easier to parse test firmware (json based or binary for example) that allows test artifacts/results to be collected in a consistant way.
This would benifit the parsing for both riotctrl, for example lots of the custom regex can be handled with a generic parser.
Some questions would be the limitations of using such an interfacem, can we still do everything we want.
Another issue might be adding complexity to testing without getting the benifits.
Ideally it would make test firmwares simpler, either way some non-trival tests should be transformed to evalutate if it makes sense.

Display Test Data Information

How do we communicate the information back from the tests to people?
Currently developers can run tests and usually get a pass fail type result.
From the CI murdock offers a go/no go situcation and errors can be viewed on failure.
What kind of information is useful (benchmarks, trends, filtering by parameters)?

xunit

xml based test result structure with that allow for pass, fail, and benchmarks of a test.
Integrates with Jenkins pretty well and has some other tools for presenting results.

webserver

We have the pull request and nightlies that display things from
murdock-html but maybe we update?

Tools to help testing

What are some tools that could help testing?

Autopost results in md format

It would be really nice to simplify uploading test results.
Maybe something as simple as posting the stdout with some env variables in hidden in the <details> wrapper to the PR from CLI.

RYOT (paper-ci)

This project wants to provide a tool for RIOT developers to easily run tests on their HW (multiple development boards).
Instead of having boards sitting on a desk or a shelf, plug in to a machine that will allow easily running tests on all those boards. Use a different machine than your personal machine, that way running tests doesn't mean halting or slowing down your own work.
The goal would be for all tests to be run on most boards at least every release, ideally every week and make this results easy to share with other RIOT developers.
Many problems with bootstapping issues.

Progress since last summit

Progress is partially tracked here , so tests can be ran on many more BOARD's.

Current Status

One machine at Inria with ~30 boards, running all tests once/per week.

- jenkins workflow to run compile_and_test_for_board.py
- githubactions to run weekly and on release candidates.
- machine is also used as a shared pool of boards among Inria Rioters (specially convenient during the

on all or a subset of the boards.
pandemic).

Experience running tests

- Too many test failures do to missing setup(privileges, driver, etc.) —> we need to filter this
- Jenkins --> great for running tests for myself
- Github actions —> great for public results
- Missing xml, missing pytest. Can we introduce pytest for all tests? pytest-integration
- Not enough time to push improvements

What is being done now

KernelCI

Linux uses
Kernal CI and we can probably learn a lot from that.
They do not want sets tied to jenkins but using jenkins.
They use internal database but want to use BigQuery for community testing.
LAVA is used? (also for Zephyr)
LAVA is a continuous integration system for deploying operating systems onto physical and virtual hardware for running tests. Tests can be simple boot testing, bootloader testing and system level testing, although extra hardware may be required for some system tests. Results are tracked over time and data can be exported for further analysis
Scalpel better than bisect to find problems...

Mbed

Uses Jenkins and now developing FPGA based board (better than PHiLIP)

IoTBench

I have no idea but will look into.

TODO:

Compile info and push to discourse sticky thread.