Building a reference IoT product with Zephyr

Ricardo Salveti
Michael Scott
Tyler Baker

Linaro connect
Budapest 2017

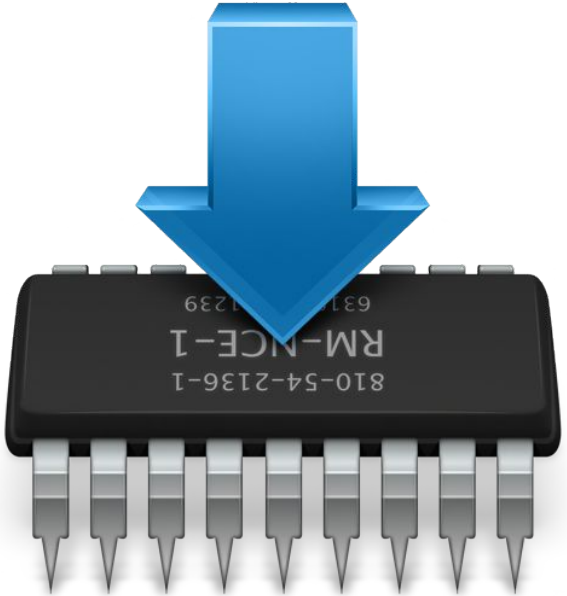
ENGINEERS AND DEVICES WORKING TOGETHER

Introduction

- Linaro Technologies
  - A small team within Linaro focusing on open source end-to-end solutions
    - Firmware to Cloud
    - IoT to Enterprise
- Who is here?
  - Ricardo Salveti
  - Michael Scott
  - Tyler Baker
- How to reach us
  - contact@linarotechnologies.org
  - https://bugs.linaro.org/describecomponents.cgi?product=Linaro%20Technologies
  - https://twitter.com/linaroorg,
  - https://twitter.com/96Boards

What are we building?

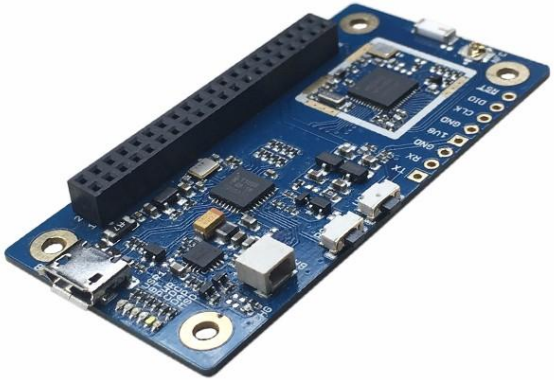
Sensor application with Firmware over the air (FOTA) support for multiple MCUs using the latest from the Zephyr project



What hardware?

96Boards Carbon
(STM32F4)

NXP K64F
(MK64FN1M0VLL12)



96Boards Nitrogen
(nRF52832)

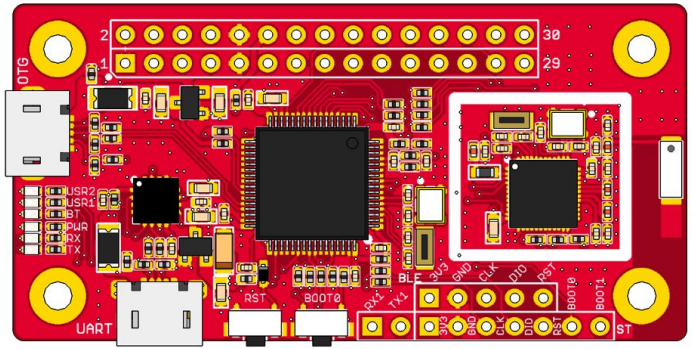
How does it work?

Project Goals

- Application
  - Publishing sensor data over a common IoT protocol (e.g. MQTT)
  - Supports delivering firmware over the air
  - A/B Partitioning Scheme, including roll back support
  - Interfaces with device management systems in the cloud
- Bootloader
  - Cryptographically validates image updates
    - Rolls back if image update fails or if not signed with the right key
    - Jumps to correct application partition if validation succeeds
- Hardware
  - Support a wide range of MCUs
- Technical Debt
  - Keep it low
    - Upstreaming platform code and keeping application changes in sync with upstream APIs
- Quality
  - Create a testable design
  - Automate all the things

Things missing when we started

- Lack of hardware support
  - MCU support missing for STM32F4 and nRF51
  - nRF51 BT LE stack provided by SoftDevice (closed implementation)
  - No support for the Cortex-M0 family
- No common bootloader support
  - No easy way to extend the boot logic out of ROM
  - Apache MyNewt's bootloader implementing most of what we wanted
- Missing drivers and subsystems support
  - Flash driver
  - SPI slave
  - No open SPI HCI firmware for nRF51



First milestone: Linaro Connect LAS16

- First FOTA done on both 96Boards Carbon and Nitrogen
  - Based on Zephyr 1.5 plus patches
  - FOTA integration using HTTP over BT LE 6LoWPAN (hawkBit), 96Boards CE as gateway
  - Using Apache MyNewt's bootloader
  - nRF51 BT LE provided as a native Zephyr application (BT LE Controller, contributed by Nordic)
- Several upstream contributions
  - ARM Cortex-M0 support
  - MCUs: nRF51, STM32F401RE
  - Boards: 96Boards Carbon, 96Boards Nitrogen
- Challenges moving forward
  - Migrate to a Zephyr based bootloader
  - Keeping the application code in sync with upstream
  - Upstream remaining changes (e.g. SPI and SPI HCI)
  - Face the substantial changes planned for Zephyr 1.6 and 1.7

Initially Zephyr was the Wild, Wild West (think before 1.5 days),
now it’s entered the Gold Rush phase of 1848 …
but it’s *still* the Wild, Wild West!

We Had Some Problems ...

Zephyr 1.6 and 1.7

Unified kernel switch

- With the move to 1.6 we had to rebase our Carbon -> nRF51 SPI driver to operate the bluetooth chip
- FOTA app fiber -> thread conversion
- Stack allocation changes

and then some more problems ...

Oh, BTW, there's a new IP stack

- TCP support wasn't functional
  - Spent 3-4 weeks debugging the issues
  - net context states were not being handled correctly for TCP
  - In some cases the local and remote IP address information wasn't being set for TCP connections
  - Incorrect callbacks for handling incoming packets
  - Lots of TCP packet sequence debugging to make sure ACK and FIN packets were handled correctly
  - TCP-specific net buf corruption when retry logic was handled via the bluetooth layer (6LoWPAN)

and then some more problems ...

Driver APIs aren't always clear

- Example: for a flash erase implementation we also needed to call the flash driver write protection (not mentioned in the API docs)

and then some more problems ...

So many knobs ...

- There are so many settings in Zephyr to tweak memory usage that it can be overwhelming
- Many of these settings don't have defaults which work with your use case
- Expect to spend time debugging the right stack sizes or # of net bufs for your app

and then some more problems ...

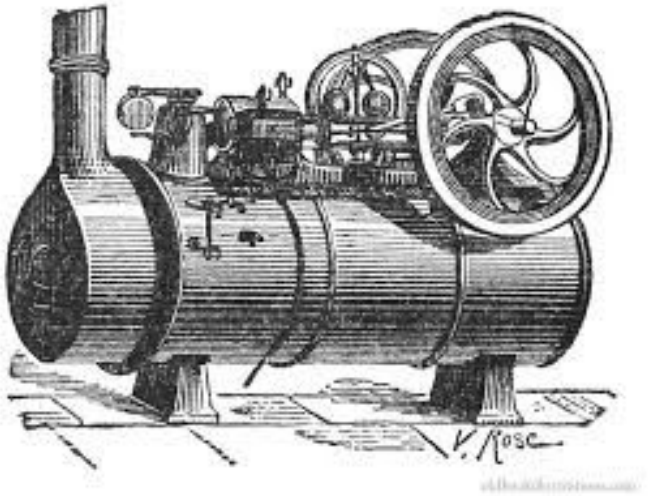
Debug/Error logs don't print unless they are enabled

- Everything is opt in at this point due to the need for smallest possible binaries.

and then some more problems ...

IPv6 support over Bluetooth Low Energy is still on early days

- Linux interface available via debugfs, not ready for production
- You can generate kernel errors by writing to this interface too quickly
- Upstream fix needs to be submitted which negative ref counts the HCI buffers



Current state of the demo (BUD17 Keynote)

- Based on Zephyr 1.7-RC3
- MCUBoot
  - Unified MyNewt based bootloader
  - Maintained outside of the Zephyr codebase
- We have got working IP
  - It will take time to get fixes upstream, and others working on the upstream may solve the problem differently, so we want to keep continuous testing and analysis going
- MQTT Integration with IBM Bluemix
  - Publishing temperature reading (MCU and external)
- Pending Zephyr changes
  - SPI Slave support (requires API changes)
  - STM32F4 SPI Master and nRF51 SPI Slave
  - SPI HCI RAW sample application (for nRF51)

Continuous Integration and Automation

- Keeping track of the sources
  - Zephyr
    - Three branches to test
      - master (upstream)
      - master-upstream-dev (upstream + linaro staged patches)
      - v1.7-dev (upstream dev branch + linaro staged patches)
  - MCUBoot
    - Two branches to test
      - master (upstream)
      - master-upstream-dev (upstream + linaro staged patches)
  - FOTA Application
    - One branch to test
      - master (upstream)
- These combination generate a matrix of permutations that constantly need to be validated
- Integrate our CI with GitHub workflow

CI Matrix for FOTA Application

Linaro Firmware Over The Air Programming Example

Example application that uses Hawkbit to implement FOTA.

Build Status:

platforms	zephyr/master	zephyr/master-upstream-dev	zephyr/v1.7-dev
96b_carbon	build passing	build passing	build passing
96b_nitrogen	build passing	build passing	build passing
frdm_k64f	build passing	build passing	build passing

Dependencies

View the build status of this project's dependencies

CI Matrix for FOTA Application Dependencies

Dependencies

View the build status of the dependent components for this project

MCU Boot Loader

Build Status

permutations	zephyr/master	zephyr/master-upstream-dev	zephyr/v1.7-dev
96b_carbon,mcuboot/master	build passing	build passing	build passing
96b_carbon,mcuboot/master-upstream-dev	build passing	build passing	build passing
96b_nitrogen,mcuboot/master	build passing	build passing	build passing
96b_nitrogen,mcuboot/master-upstream-dev	build passing	build passing	build passing
frdm_k64f,mcuboot/master	build passing	build passing	build passing
frdm_k64f,mcuboot/master-upstream-dev	build passing	build passing	build passing

Zephyr

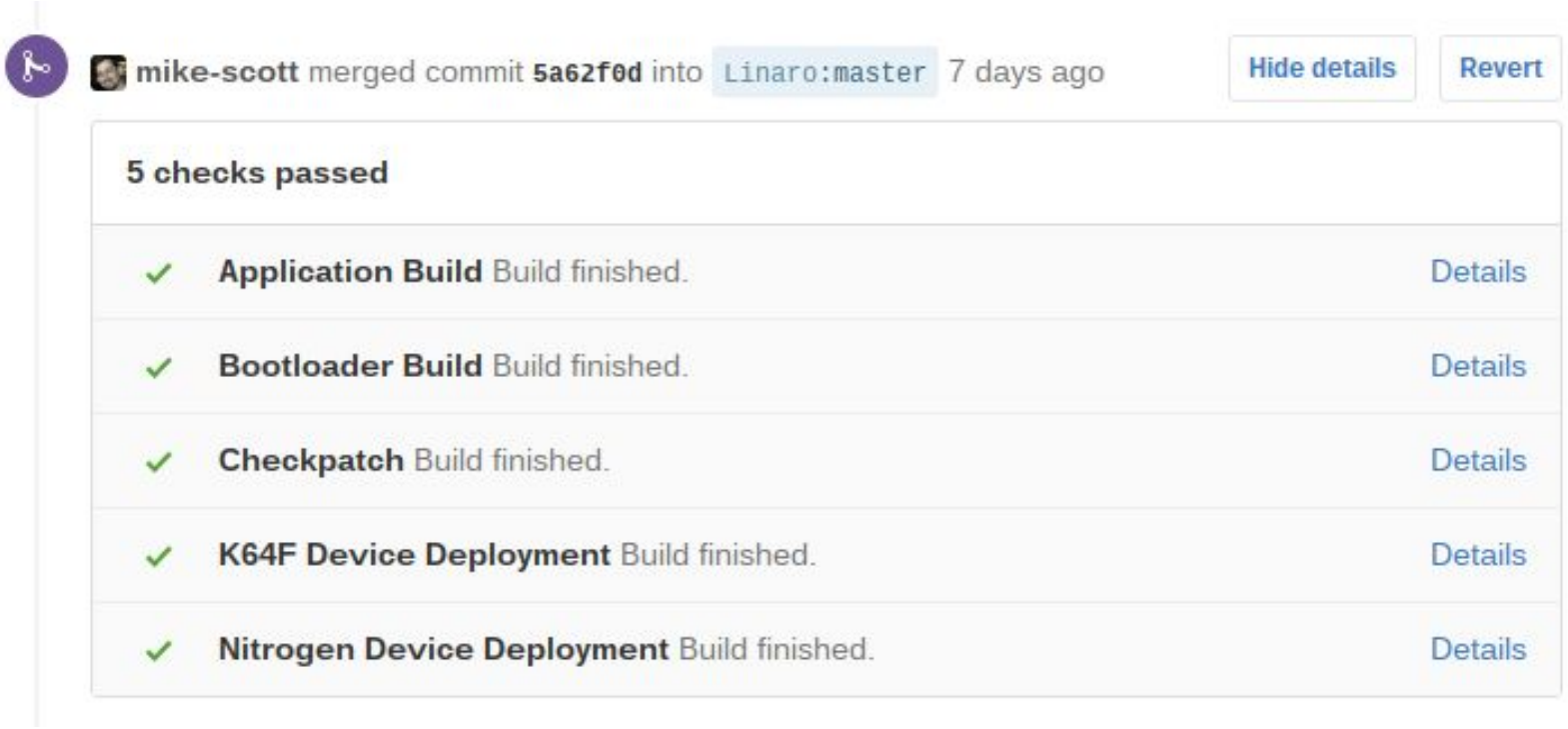
Build Status

platform	zephyr/master	zephyr/master-upstream-dev	zephyr/v1.7-dev
96b_carbon	build passing	build passing	build passing
96b_nitrogen	build passing	build passing	build passing
frdm_k64f	build passing	build passing	build passing
qemu_cortex_m3	build passing	build passing	build passing

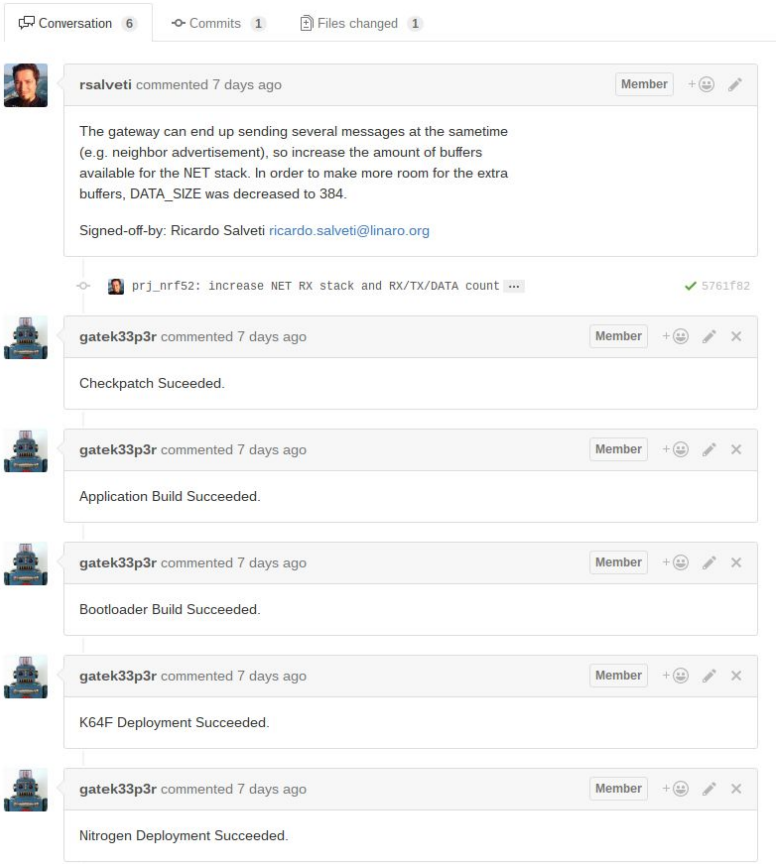
Continuous Integration and Automation Continued

- Strategies
  - Question that needed to be answered
    - How can we “stay close to upstream”
    - How can we “reduce any/all technical debt”
    - How can we do all this and still produce something stable
    - How do we enable scalable deployment for real life products
  - Our solutions
    - For each project, on each branch, on every single merge we automatically do:
      - Build tests
      - Run unit tests (Zephyr test applications) on supported hardware
      - Run functional tests on our application of supported hardware
      - Test end to end device update functionality
  - How does this help keep us sane?
    - We know the moment when a build, unit test, or functional tests fails upstream
      - Allows us to locate a fix before we rebase our dev branch
      - Bisect problems quickly

Pre Merge Testing



Pre Merge Testing



Hardware Testing

- LAVA
  - Developed and upstreamed a ‘bare metal’ testing interface (lava-test-monitor)
    - Detects start and end test cases
    - Parses console output
    - Can send commands, and parse the command output
  - New IoT device support
    - Firmware Tools
      - Pyocd
      - Dfu-utils
      - Mass-storage
    - Devices
      - Nitrogen
      - Carbon
      - K64F
      - Arduino101
      - V2m-beetle
      - Nucleo-l476rg
      - QEMU (x86, cortex-m3)

Hardware Testing Continued

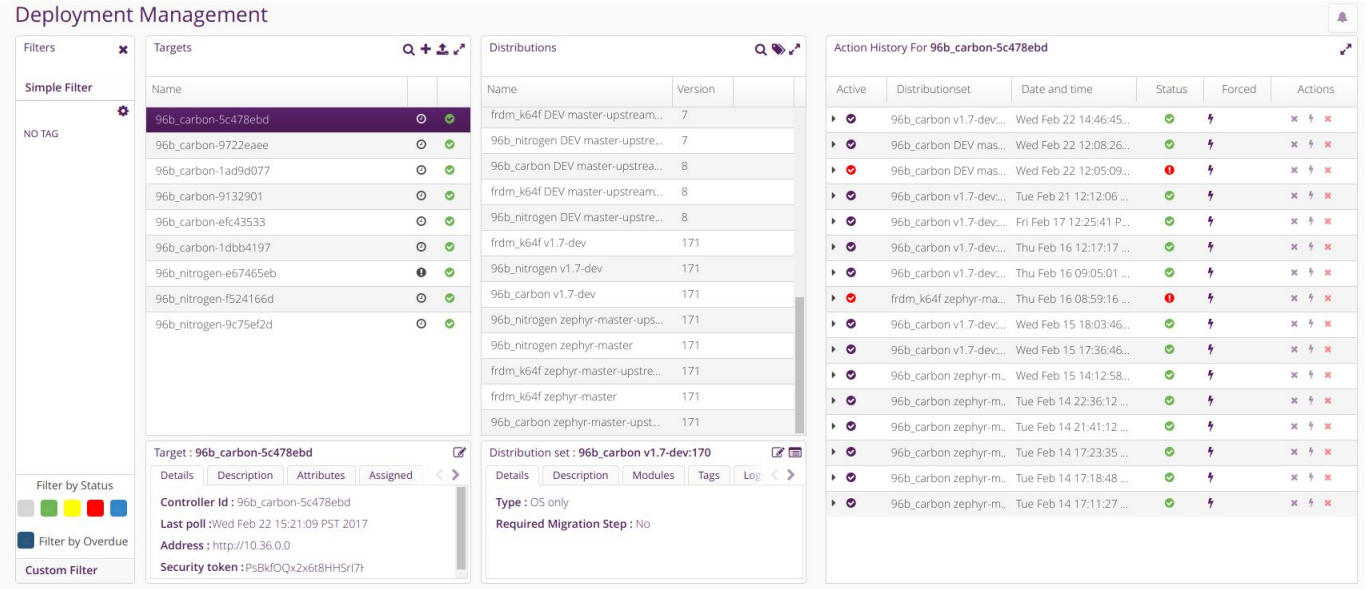
LAVA | Home | Dashboard | Results | Scheduler | API | Help | Instance: lava-docker-instance | admin

Copy to clipboard

```
device_type: nrf52-nitrogen
job_name: nitrogen-github-pr-55-fota-hawkbit

timeouts:
  job:
    minutes: 6
  action:
    minutes: 3
priority: medium
visibility: public

actions:
- deploy:
    timeout:
      minutes: 3
    to: tmpfs
    images:
        bootloader:
          image_arg: '-ce -a 0x0 {bootloader}'
          url: http://192.168.1.17/pr-linaro-fota-hawkbit/55/mcuboot/96b_nitrogen/zephyr.bin
        fota:
          image_arg: '-a 0x8000 {fota}'
          url: http://192.168.1.17/pr-linaro-fota-hawkbit/55/linaro-fota-hawkbit/96b_nitrogen/zephyr-signed.bin

- boot:
    method: pyocd
    timeout:
      minutes: 5

- test:
    monitors:
    - name: Linaro FOTA Hawkbit Test
      start: tc_start()
      end: PROJECT EXECUTION
      pattern: (?P<result>(PASS|FAIL))\s-\s(?P<test_case_id>\w+)
      fixupdict:
        PASS: pass
        FAIL: fail
```

Rollout Testing

- Hawkbit
  - CI Builds are pushed into Hawkbit
  - Rolled out to devices once a canary build is flashed and validated by LAVA
  - Automated via Hawkbit's REST API



Developer Testing

- Allows developers to trigger ‘custom’ CI build and test jobs
  - Simply provide a git urls and branches
  - Runs sanitycheck, builds FOTA application, etc..
  - Useful for rebase testing and ‘crazy’ ideas
- Future Plans
  - Developer triggered device testing
  - Automated rollouts for a common pool of devices distributed around the world
  - Deliver bootloader over the air

Thank You, Any Questions?

#BUD17

For further information: www.linaro.org

BUD17 keynotes and videos on: connect.linaro.org