Linaro

ELC 2017 Zephyr

Riding the upstream wave: Keeping your Zephyr applications regression free

February 2017

LEADING COLLABORATION IN THE ARM ECOSYSTEM

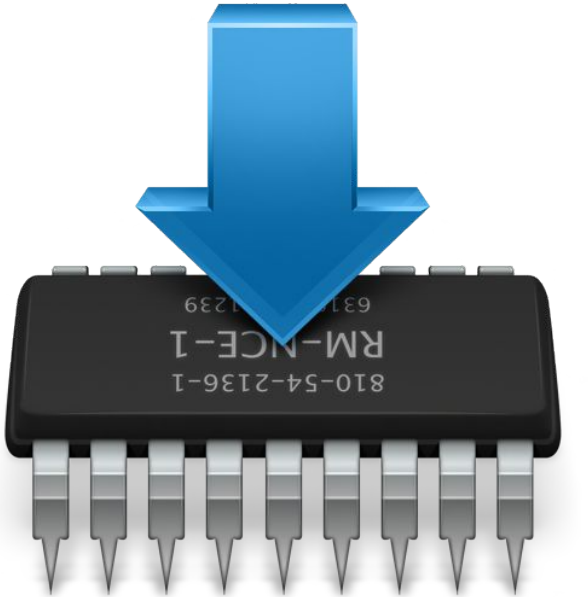
Linaro TECHNOLOGIES

Introduction

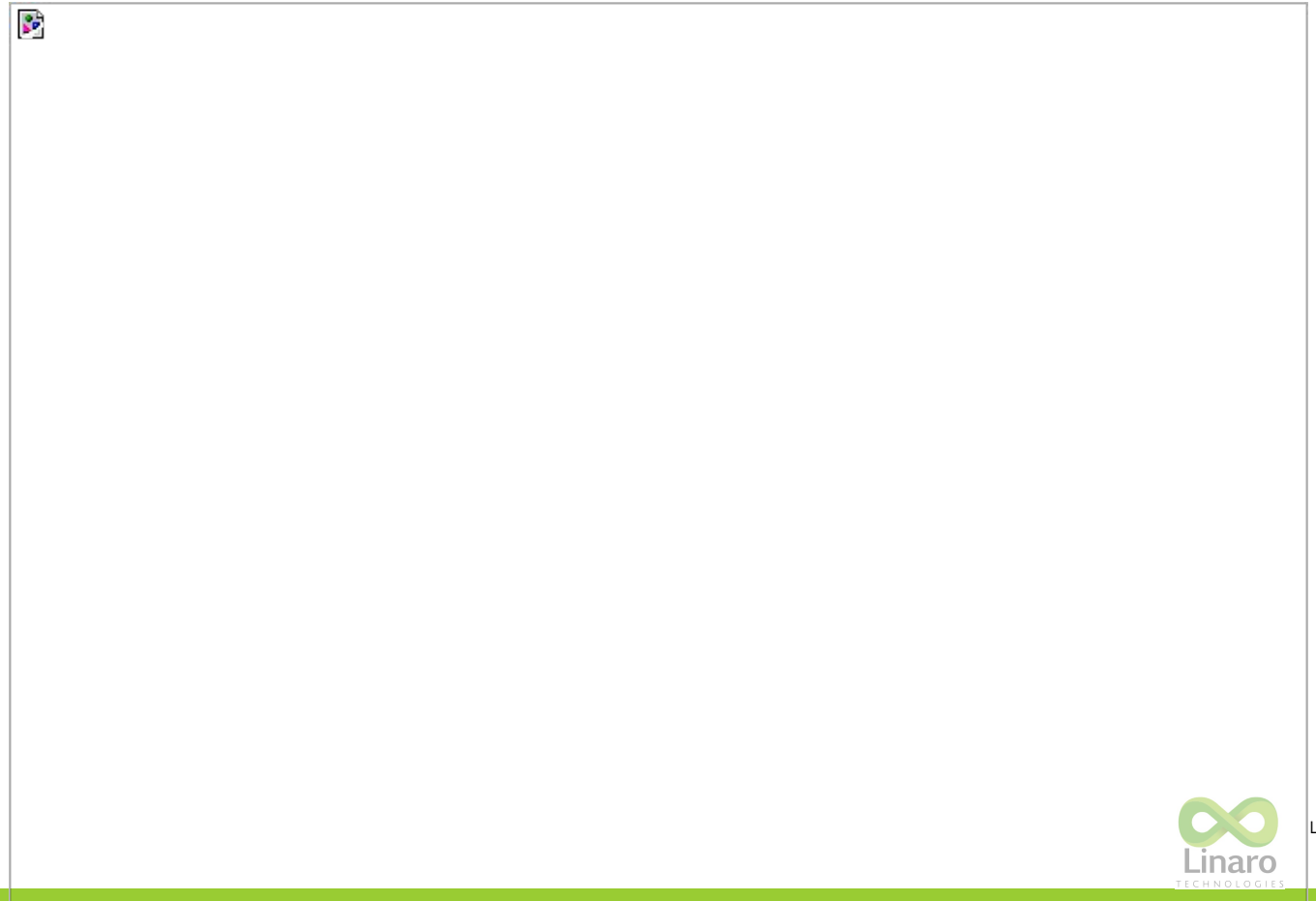
- Linaro
  - Collaborative engineering organization consolidating and optimizing open source software and tools for the ARM architecture.
- Technologies Division
  - Focusing on open source solutions for real world problems.
    - Firmware to Cloud
    - IoT to Enterprise
- Engineers
  - Ricardo Salveti
  - Michael Scott
  - Tyler Baker

What are we building?

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



How does it work?



Project Goals

- Application
  - Supports delivering firmware over the air
  - Partitions flash, and writes firmware downloaded to flash
  - Interfaces with device management systems in the cloud
- Bootloader
  - Cryptographically validates image updates
    - Rolls back if image update fails
    - Jumps to correct application partition if validation succeeds
- Hardware
  - Support a wide range of MCUs
- Technical Debt
  - Keep it low
    - Upstreaming platform code
    - Keeping application changes in sync with upstream APIs
- Quality
  - Create a testable design
  - Automate all the things

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!

There are some large problems...

- Zephyr 1.6 and 1.7
  - New IP stack isn't completely done.
    - Much of the more popular functionality works to some degree, but expect bugs. TCP via bluetooth 6lowpan in particular still needs work.
  - So many knobs.
    - Expect to spend time debugging the right stack sizes or # of net bufs because the defaults aren't optimal or they don't apply to your use case.
  - 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.
  - IPv6 support over Bluetooth Low Energy is still on early days.
    - Linux interface available via debugfs, and not yet used on production.

Internet Protocol and Bootloader

- We’ve gotten IP to work
  - 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 analysis going
- MCUBoot
  - Maintained in a separated tree (like a normal zephyr application), so API incompatibilities with Zephyr is expected

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

Continuous Integration and Automation

- 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
  - 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