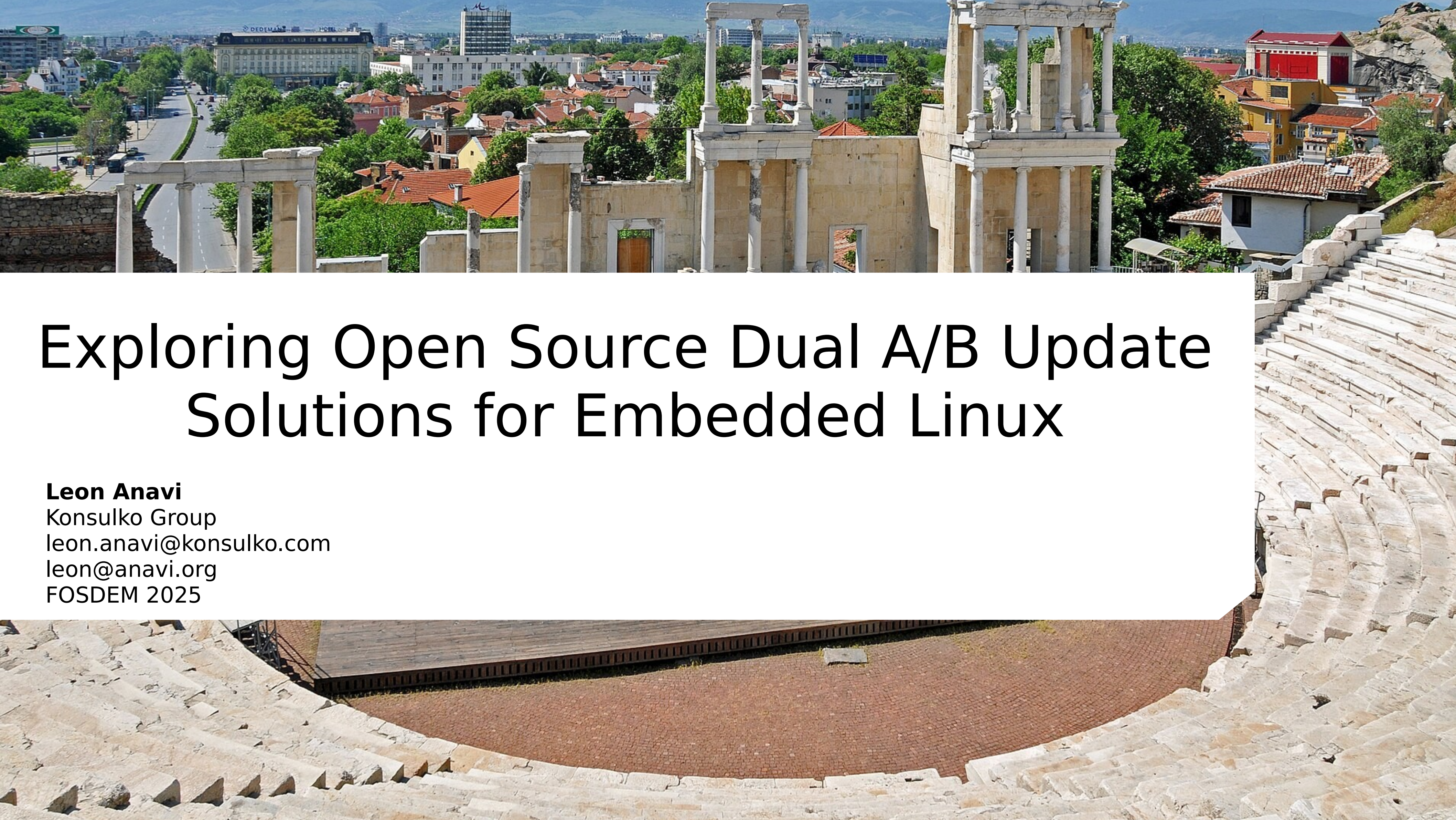


Exploring Open Source Dual A/B Update Solutions for Embedded Linux

Leon Anavi
Konsulko Group
leon.anavi@konsulko.com
leon@anavi.org
FOSDEM 2025

Agenda

- Embedded Linux update strategies and open source solutions
- Mender
- RAUC
- SWUpdate
- Conclusions

Common Embedded Linux Update Strategies

- A/B updates (dual redundant scheme)
- Delta (or adaptive) updates
- Container-based updates
- Combined strategies

A/B Updates

- Dual A/B identical rootfs partitions
- Data partition for storing any persistent data which is left unchanged during the update process
- Typically a client application runs on the embedded device and periodically connects to a server to check for updates
- If a new software update is available, the client downloads and installs it on the other partition
- The bootloader switches the active partitions on reboot aftre upgrade
- Fallback in case of update failure

Delta Updates

- Only the binary delta between the difference is sent to the embedded device
- Works in a Git-like model for filesystem trees
- Saves storage space and connection bandwidth
- Rollback of the system to a previous state

Side by Side Comparison

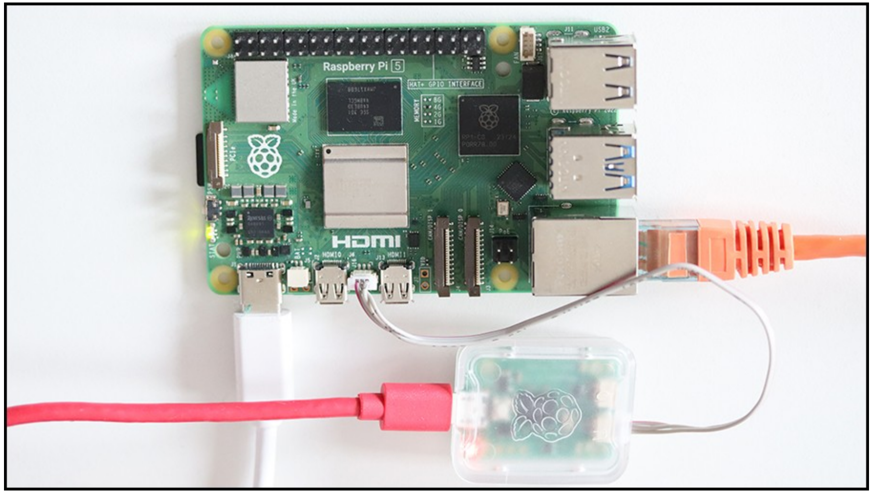
Update Strategy	Storage Space	Update Size	Rollback to a Previous Stage	Fallback to a Back-up Image on a separate partition
A/B Updates	Large	Large	Yes	Yes
Delta Updates	Small	Small	Yes	No

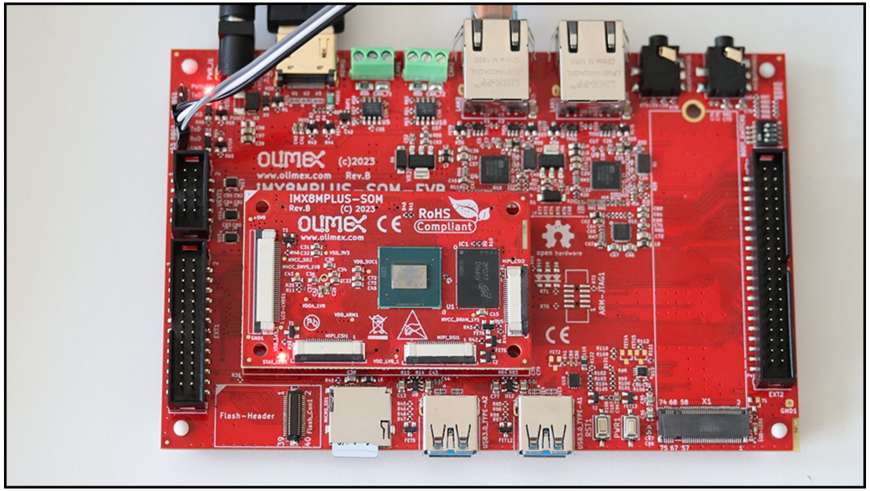
Popular Open Source Solutions

- **Mender**
- **RAUC**
- **SWUpdate**
- Swupd
- UpdateHub
- Balena
- Memfault
- qbee.io
- Snap
- Libostree (OSTree)
- Flatpak
- QtOTA
- Torizon
- Aktualizr-lite
- HERE OTA Connect (Aktualizr) ❌
- FullMetalUpdate ❌

Side by Side Comparison Using

- Raspberry Pi 5
- Olimex iMX8MP-SOM-4GB-IND and iMX8MP-SOM-EVB-IND



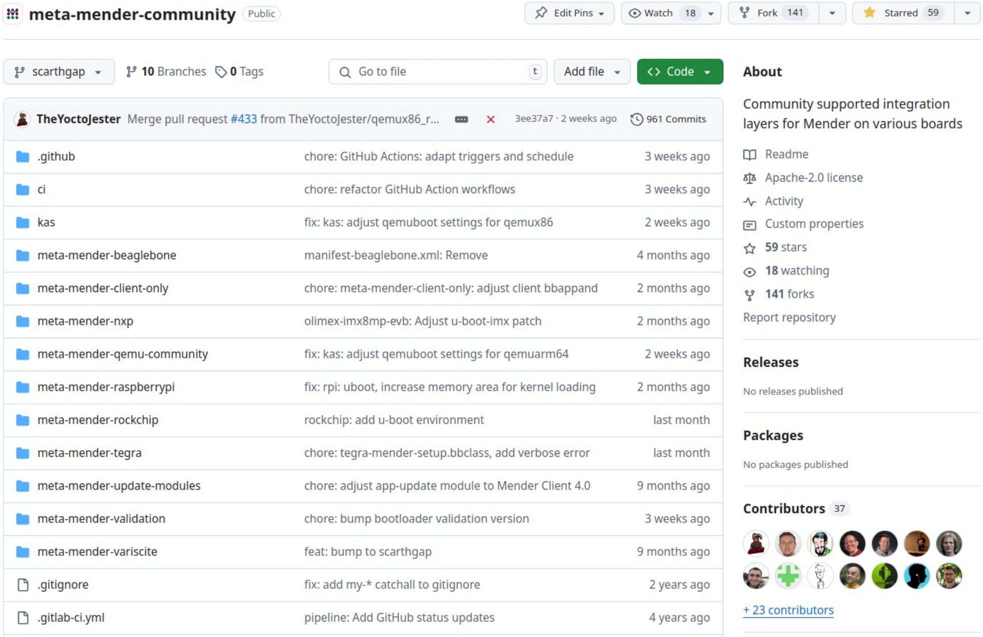


Mender

- Available as a free open source or paid commercial/enterprise plans
- **A/B** update scheme for open source users and all plans as well as **delta** updates for professional and enterprise plans
- Back-end services (Hosted Mender)
- Written in C++, Go, Python, JavaScript
- Source code in GitHub under Apache 2.0
- Supports the Yocto Project and Debian family of Linux distributions

Mender Supported Devices

- Raspberry Pi
- Rockchip
- BeagleBone
- x86-64
- NXP
- NVIDIA Tegra
- QEMU



TheYoctoJester Merge pull request #433 from TheYoctoJester/qemux86_r... 3ee37a7 · 2 weeks ago 961 Commits

.github	chore: GitHub Actions: adapt triggers and schedule	3 weeks ago
ci	chore: refactor GitHub Action workflows	3 weeks ago
kas	fix: kas: adjust qemuboot settings for qemux86	2 weeks ago
meta-mender-beaglebone	manifest-beaglebone.xml: Remove	4 months ago
meta-mender-client-only	chore: meta-mender-client-only: adjust client bbappand	2 months ago
meta-mender-nxp	olimex-imx8mp-evb: Adjust u-boot-imx patch	2 months ago
meta-mender-qemu-community	fix: kas: adjust qemuboot settings for qemuarm64	2 weeks ago
meta-mender-raspberrypi	fix: rpi: uboot, increase memory area for kernel loading	2 months ago
meta-mender-rockchip	rockchip: add u-boot environment	last month
meta-mender-tegra	chore: tegra-mender-setup.bbclass, add verbose error	last month
meta-mender-update-modules	chore: adjust app-update module to Mender Client 4.0	9 months ago
meta-mender-validation	chore: bump bootloader validation version	3 weeks ago
meta-mender-variscite	feat: bump to scarthgap	9 months ago
.gitignore	fix: add my-* catchall to gitignore	2 years ago
.gitlab-ci.yml	pipeline: Add GitHub status updates	4 years ago

About

Community supported integration layers for Mender on various boards

Readme
Apache-2.0 license
Activity
Custom properties
59 stars
18 watching
141 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 37

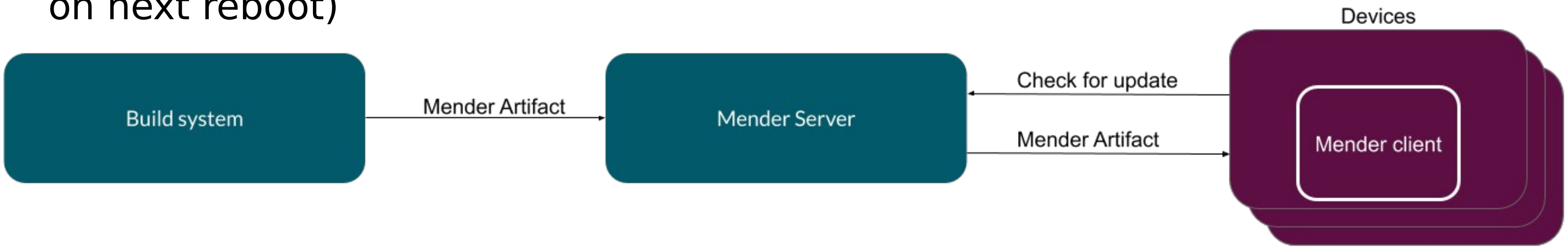
+ 23 contributors

- Details: https://github.com/mendersoftware/meta-mender-community

Mender

Steps to install Mender A/B update on embedded Device:

- Apply update
- Reboot
- On the first boot after a successful update, though the Mender client a commit must be performed to accept the update (otherwise the system will roll-back on next reboot)



Mender Client Modes

Mender A/B updates supports two client modes:

- Managed (default) - client running as a daemon polls the server for updates
- Standalone - updates are triggered locally which is suitable for physical media or any network update in pull mode

```
SYSTEMD_AUTO_ENABLE:pn-mender = "disable"
```

```
$ cd tmp/deploy/images/raspberrypi5
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
$ mender -install http://example.com:8000/core-image-base-raspberrypi5.mender
```

Mender Classes and Variables

- Inherit Mender classes globally:

```
INHERIT += "mender-full"
```

- Mender uses specific variables during the build process:

```
local_conf_header:
  olimex-imx8mp-evb: |
    MENDER_IMAGE_BOOTLOADER_FILE = "imx-boot"
    MENDER_IMAGE_BOOTLOADER_BOOTSECTOR_OFFSET = "64"
    MENDER_UBOOT_STORAGE_INTERFACE = "mmc"
    MENDER_UBOOT_STORAGE_DEVICE = "1"
    MENDER_STORAGE_DEVICE = "/dev/mmcblk1"
    IMAGE_BOOT_FILES:append = "boot.scr"
```

Mender Data Partition

- Mender creates a **/data** partition to store persistent data, preserved during Mender updates. Supports ext4, Btrfs and F2FS file systems.
- The Mender client on the embedded devices uses **/data/mender** to preserve data and state across updates
- Variable **MENDER_DATA_PART_SIZE_MB** configures the size of the **/data** partition. By default it is 128 MB. If enabled, mender feature **mender-growfs-data** which relies on **systemd-growfs** tries to resize on first boot with the remaining free space
- It is possible to create an image for the data partition in advance with bitbake:

```
IMAGE_FSTYPES:append = " dataimg"
```

Mender add-ons

Mender supports several add-ons:

- **Remote Terminal** - interactive shell sessions with full terminal emulation
- **File Transfer** - upload and download files to and from a device
- **Port Forward** - forward any local port to a port on a device without opening ports on the device
- **Configure** - apply configuration to your devices through a uniform interface

Mender Delta Updates

- Mender offers robust delta update rootfs as a module for the **commercial** Mender plan (**closed source** implementation)
- Requires reboot to apply the update
- Supports rollback
- mender-binary-delta creates a binary delta by comparing two different artifacts
- Mandatory requirement for the implementation is a **read-only** root file system:

```
IMAGE_FEATURES += "read-only-rootfs"
```

```
EXTRA_IMAGE_FEATURES = "read-only-rootfs"
```

RAUC

- A lightweight update client that runs on an Embedded Linux device and reliably controls software A/B updates
- Supports multiple update scenarios
- Supports HTTP streaming and adaptive updates
- Provides tool for the build system to create, inspect and modify update bundles
- Uses X.509 cryptography to sign update bundles
- Supports encrypted update bundles
- Compatible with the Yocto Project, PTXdist and Buildroot

RAUC Licenses

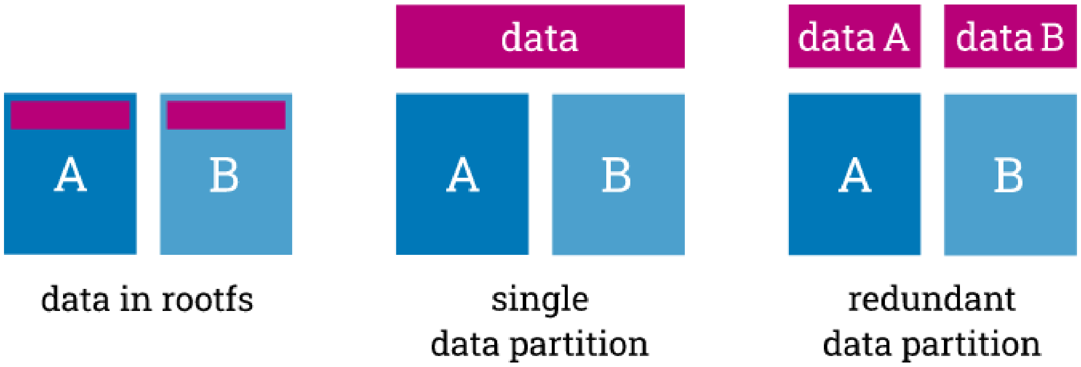
- RAUC – LGPLv2.1
  https://github.com/rauc/rauc
- meta-rauc - MIT
  https://github.com/rauc/meta-rauc
- rauc-hawkbit – LGPLv2.1
  https://github.com/rauc/rauc-hawkbit
- rauc-hawkbit-updater – LGPLv2.1
  https://github.com/rauc/rauc-hawkbit-updater

RAUC Integration Steps

- Select an appropriate bootloader
- Enable **SquashFS** in the Linux kernel configurations
- **ext4** root file system (RAUC does not have an ext2 / ext3 file type)
- Create specific partitions that matches the RAUC slots in the OpenEmbedded Kickstart (.wks) file
- Configure Bootloader environment and create a script to switch RAUC slots
- Create a certificate and a keyring to RAUC’s **system.conf**

RAUC Data Partition

- Supports single and redundant data partitions
- For redundant data partitions the active rootfs slot has to mount the correct data partition dynamically, for example with a udev rule



RAUC Advanced Features

- **HTTP Streaming**

  Supports installing bundles directly from a HTTP(S) server, without having to download and store the bundle locally

- **Adaptive Updates**

  Adaptive updates can be installed on any version, using data from the target system, such as previous versions or even interrupted installations. Paired with **HTTP Streaming**, RAUC downloads only the required parts of the bundle, improving efficiency.

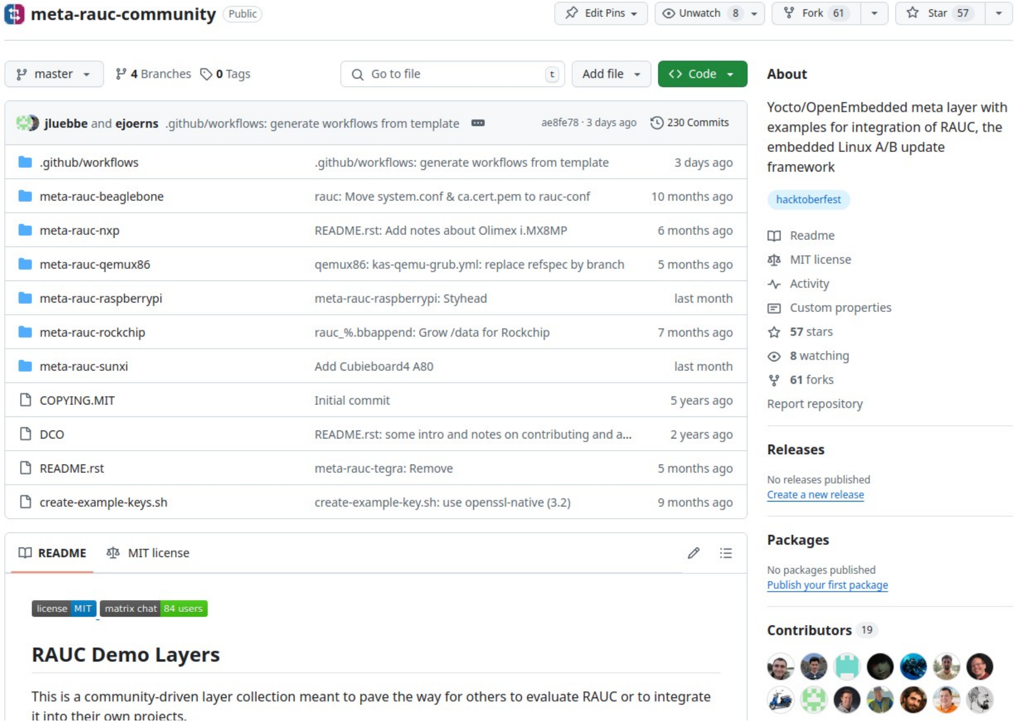
meta-rauc-community

- Yocto/OE layer with examples how to integrate RAUC on various machines
- Started in 2020
- Moved to the RAUC organization in GitHub in 2021
- https://github.com/rauc/meta-rauc-community/

Contributions are always welcome as GitHub pull requests!

meta-rauc-community

- Raspberry Pi
- BeagleBone
- x86-64
- NXP
- QEMU
- Rockchip
- Allwinner (Sunxi)
- DHSBC STM32MP13



jluebbe and ejoerns .github/workflows: generate workflows from template ae8fe78 · 3 days ago 230 Commits

.github/workflows	.github/workflows: generate workflows from template	3 days ago
meta-rauc-beaglebone	rauc: Move system.conf & ca.cert.pem to rauc-conf	10 months ago
meta-rauc-nxp	README.rst: Add notes about Olimex i.MX8MP	6 months ago
meta-rauc-qemux86	qemux86: kas-qemu-grub.yml: replace refspec by branch	5 months ago
meta-rauc-raspberrypi	meta-rauc-raspberrypi: Styhead	last month
meta-rauc-rockchip	rauc_%.bbappend: Grow /data for Rockchip	7 months ago
meta-rauc-sunxi	Add Cubieboard4 A80	last month
COPYING.MIT	Initial commit	5 years ago
DCO	README.rst: some intro and notes on contributing and a...	2 years ago
README.rst	meta-rauc-tegra: Remove	5 months ago
create-example-keys.sh	create-example-key.sh: use openssl-native (3.2)	9 months ago

RAUC Demo Layers

This is a community-driven layer collection meant to pave the way for others to evaluate RAUC or to integrate it into their own projects.

About

Yocto/OpenEmbedded meta layer with examples for integration of RAUC, the embedded Linux A/B update framework

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 19

SWUpdate

- A flexible open source update framework with small footprint for atomic updates
- Supports signing with RSA keys and with certificates using an own PKI infrastructure
- Supports incremental update of binary images
- Supports Lua extensions
- Compatible with the Yocto Project, Buildroot and deb package (experimental)

SWUpdate

- SWUpdate under GPLv2
- A library to control SWUpdate under LGPLv2.1.
- Extensions written in Lua under Lua license (MIT)
- Supports the Yocto Project / OpenEmbedded and Debian / Ubuntu
- Supported devices through Yocto/OE layer meta-swupdate-boards:

  Beaglebone Black, Raspberry Pi, Sama5d27-som1-ek-sd and Wandboard

Side by Side Comparison

Features	Mender	RAUC	SWUpdate
A/B updates	Yes	Yes	Yes
Roll-back	Yes	Yes	Yes
Configure add-on	Yes	No	No
Monitor add-on	Yes	No	No
Troubleshot add-on	Yes	No	No
Local web interface	No	No	Yes

Side by Side Comparison

Features	Mender	RAUC	SWUpdate
Client Programming Language	C++ (previously Go)	C	C
Client License	Apache 2.0	LGPL-2.1	GPLv2
Yocto Project Integration	Scarthgap	Scarthgap	Scarthgap
Contributions	GitHub Pull Requests	GitHub Pull Requests	Mailing List
Management Server	Yes	3rd Party	3rd Party

3rd Party Management Servers

- **Eclipse HawkBit**

  https://eclipse.dev/hawkbit/

- **qbee.io**

  https://github.com/qbee-io/meta-qbee

- **AWS IoT**

  https://github.com/aws4embeddedlinux/meta-aws

libubootenv

- Provides a hardware independent way to access to U-Boot environment
- Includes replacements for the "fw_printenv" and "fw_setenv" tools, which are compatible with any board
- Written in C
- Available in GitHub under LGPL-2.1
- Started by Stefano Babic in December 2018
- Used by SWUpdate, RAUC, Mender and other solutions
- OpenEmbedded/Yocto recipe: https://git.openembedded.org/openembedded-core/tree/meta/recipes-bsp/u-boot/

Combined Strategies with Containers

- Yocto/OE layer **meta-virtualization** provides support for building Xen, KVM, Libvirt, docker and associated packages necessary for constructing OE-based virtualized solutions
- **virtualization** has to be added to the **DISTRO_FEATURES**:

```
DISTRO_FEATURES:append = " virtualization"
```

- For example adding Docker to the embedded Linux distribution is easy:

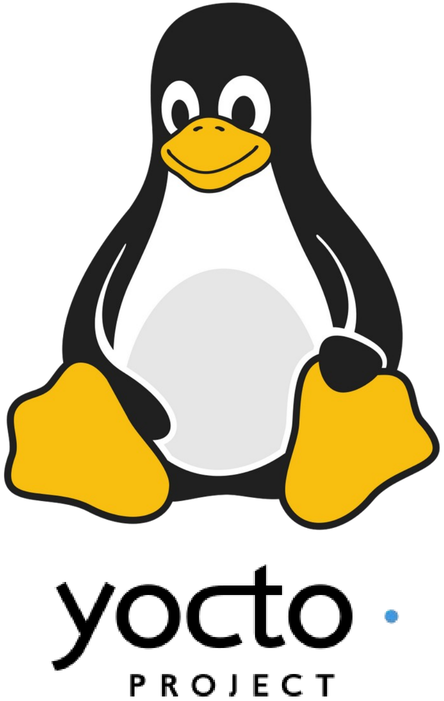
```
IMAGE_INSTALL:append = " docker-moby"
```

- There are use cases on powerful embedded Linux devices where containers are combined with A/B updates of the base Linux distribution built with Yocto/OE

Conclusions

- With many reliable open-source solutions available for updating embedded Linux devices, developing an in-house solution is rarely worth the effort
- The dual A/B update mechanism implementation depends on the bootloader
- Mender, RAUC, and SWUpdate all handle A/B updates effectively but differ in how they are implemented and the advanced features they offer
- Mender provides an end to end turn-key solution with management server
- Delta and/or adaptive updates are also possible with Mender and RAUC
- Choosing the best solution can be challenging, as it depends on the specific requirements of your project

Thank You!



Useful links:

- https://www.yoctoproject.org/
- https://mender.io/
- https://rauc.io/
- https://swupdate.org/
- https://www.konsulko.com/mender-raspberry-pi-5
- https://www.konsulko.com/ota-updates-imx8mp-mender
- https://www.konsulko.com/ota-qbee-rauc-imx8mp