Silicon Blade Consultants

EMBEDDED LINUX · YOCTO · TRAINING · CONSULTING

Getting OTA Working on Zephyr

It Can't Be That Difficult, Can It?

Zephyr Project Meetup – November 2026

Iain Menzies-Runciman
Consultant and Trainer

ENGINEERING KNOWLEDGE INTO REAL PRODUCTS

About me

Ego

Over 30 years of experience working with Linux and embedded systems

Specialising in Yocto, Embedded Linux, BSP development and system integration

Helping companies develop, debug, deploy and maintain embedded Linux products

Passionate about practical engineering and understanding **why things work**, not just how



The cunning plan

Simple on paper...

- Take a familiar embedded Linux OTA workflow
  - Mender OTA updates
- Apply them to Zephyr on a microcontroller
- Install to an inactive slot
- Boot, confirm and retail rollback

Starting point

Gap analysis

What I knew

- Embedded Linux
- Mender client
- A/B root filesystems
- Bootloader integration

What would be new

- Zephyr
- MCUboot slots
- Sysbuild
- Highly constrained memory

First attempt: ESP32

What have I got lying around

- Random ESP32 from an old device

- Start simple
  - Zephyr Wi-Fi example

- Complify
  - Networking
  - DNS
  - TLS

Zephyr modules

Start learning

- My `west.yml` was similar to my kas configuration files
  - Add the repos
  - define the paths

- The `prj.conf` file is my `.config` file
  - `CONFIG_SOME_OPTION=value`

This is going to be simple...

TLS

The first detour

- Needed to enable `CONFIG_MBEDTLS_PSA_CRYPTO_C`
  - Which brought in additional dependencies
  - Which were not configured
  - So the build failed

- Fixed the dependencies

It compiled...

The the ESP32 ran out of room

Too used to MPUs

After adding

- Wi-Fi network stack
- TLS and network buffers
- Application and system stacks
- Mender

I had run out of RAM

Plan B

If at first you don’t succeed – change the story

I had just got my hands on a new board

- FRDM-MCXN947

This was a good chance to play!

FRDM-MCXN947

Changes...

What changed

- Ethernet not Wi-Fi
- More memory headroom
- Different drivers for the board

What stayed the same

TLS and identity

Mender authentication

Persistent identity

No ID cards available...

- Mender needs an identity that does not change
  - Often uses the MAC address

- I was getting random addresses
  - Good to stop tracking
  - Bad as Mender needs to track

- Fixed the MAC address

Ethernet issues

Just because it exists does not mean it is useable

- Fixed the MAC address
- Had the correct driver
- “No new RX buf available” meant buffer starvation
  - Increased the buffers
  - Waited for the PHY link before activating

Achievement unlocked

Each step takes you closer

TERMINAL / CODE

```
Mender authentication accepted
```

The device could authenticate, but it still could not install firmware.

A Valid artifact is not enough

Continue

First attempt

- Package zephyr.bin
- Upload to Mender
- Server accepts artifact

But no install mechanism

What was missing

- MCUboot
- An inactive firmware slot
- Zephyr image update module
- Signed application image

Sequence of events needed

Architecture



Artifact changes

Security matters

zephyr.bin

↓

zephyr.signed.bin

Mender payload type: zephyr-image

MCUBoot

An instructive failure

TERMINAL / CODE

```
Image in the primary slot is not valid!
Unable to find bootable image
```

A plain application-only build was no longer a valid baseline once MCUboot owned the boot process.

Memory problems. Again

Need to optimise. Again

Before

- ~281 KiB main SRAM usage
- 128 KiB mbedTLS heap
- Oversized network pools
- Shared workqueue

After

- ~92 KiB main SRAM usage
- 64 KiB TLS heap in SRAMX
- Right-sized network pools
- Dedicated Mender workqueue

Deployment polling

Finding the limits

TERMINAL / CODE

```
Stack overflow: k_sys_work_q

CONFIG_MENDER_SCHEDULER_SEPARATE_WORK_QUEUE=y
CONFIG_MENDER_SCHEDULER_WORK_QUEUE_STACK_SIZE=6
```

Saving a few KiB is not an optimisation if the shared stack then overflows.

The rest of the network matters

Nothing in isolation

- Artifact URL resolved to `s3.docker.mender.io`
  - But I was self-hosting for testing
- Self-hosted API access did not prove artifact access
- DNS and service configuration needed attention
- Socket/file descriptor limits also needed increasing

Complete update

Finally!



If the new image fails before confirmation, MCUboot can roll back.

Lessons learnt

If I were to do over

- There is less wriggle room on MCUs
- Set MCUboot and flash layout first
- Budget memory by region, not just total RAM
- Establish stable device identity early

Working demo is not production

This is not safe for work!

What works so far

- Mender deployment
- Signed image installation
- Test boot and confirmation
- Rollback capability

Still to address

- Proper hardware entropy source
- Production signing keys
- Full security review
- Failure and recovery testing

Questions?

I may not know the answer though

Source and implementation:

github.com/ming4real/mender-zephyr-demo

www.siliconbladeconsultants.com/articles/20260830-zephyr-mender/