# Combining white-labeled hardware with Open Source software

Nick Chen, Blecon

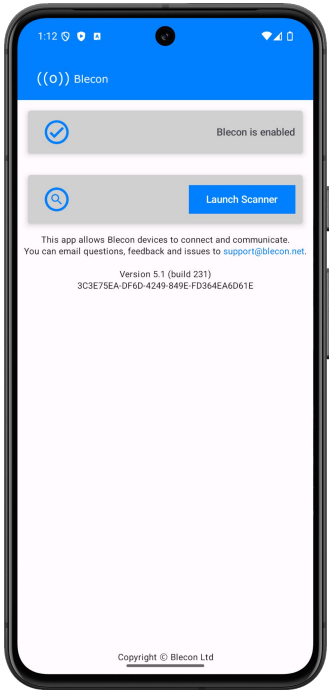
# Problem: we needed a dedicated hardware hotspot

**Hardware products and startups can be a difficult match**

Unique skills required

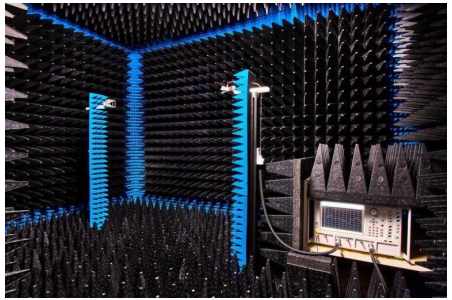
Longer time scales

Upfront costs and challenging scaling curves



# A prototype is only the tip of the iceberg

Prototype



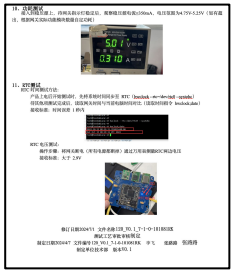
Certification

Verification Tests

Manufacturing tools

Mechanical design

Mechanical tooling

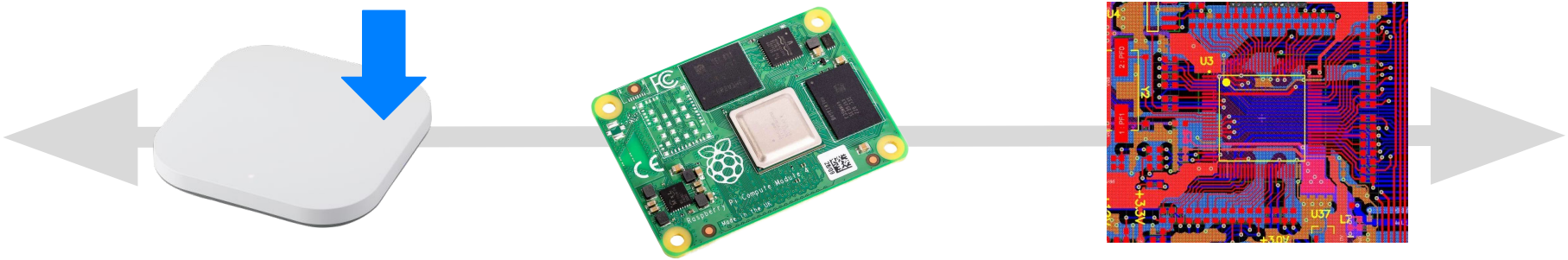


Test plans

Inventory

# A spectrum of options simplifies this process

*"Hardware Only"* White Label



**White label**

**Modular hardware**
Mechanical design
Manufacturing

**Fully custom**
Electronics
Mechanical design
Certification
Manufacturing

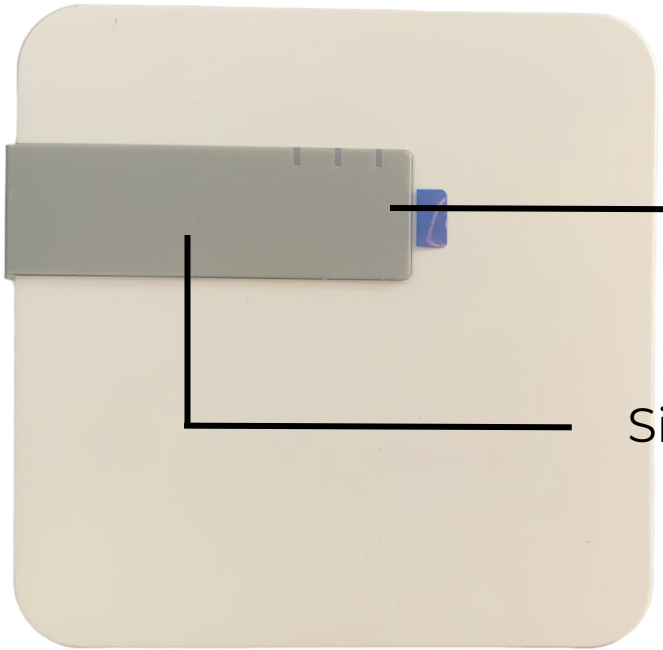
# Leveraging the ecosystem’s strengths

Economies of scale

Certification/Verification Testing

Manufacturing experience

# White label offers customizability



Plastics

Silkscreen

Laser Engraving

# Gaps in the pure white label approach

| Area | Requirement |
| --- | --- |
| Software requirements | Low-level access to Bluetooth hardware, resiliency |
| User experience | Plug-and-play with cloud-based config |
| Maintainability | Ability to develop without vendor expertise/support |
| Security | Ensuring that no ports or debug interfaces were exposed, regular cadence of security updates |

# Example costs and timelines

**Off The Shelf**

Firmware validation: 3 weeks

Check samples: 1 weeks

Mass production: 3 weeks

NRE waived with order quantity

**Customized Hardware Adds:**

Hardware validation: 12 weeks

Certification: 8 weeks

$15K - $25K NRE

# Rest of this Presentation

**Tips For**

1. Setting up
2. Developing
3. Manufacturing

Things I did that paid off

Things I wish I had known about

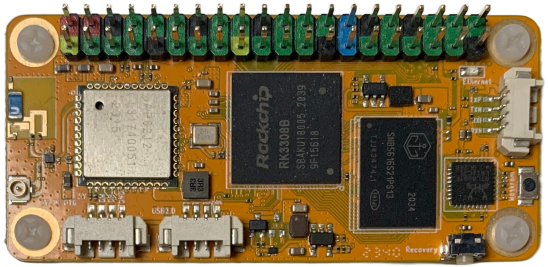
Things that happened

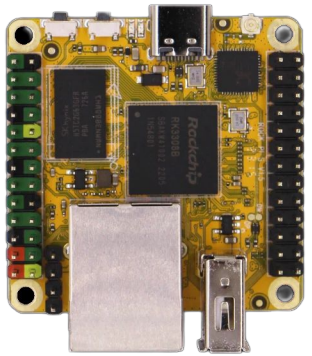
# 1. Initial Preparation

# Hardware Documentation Can Be Sketchy

**Look for better-supported platforms**

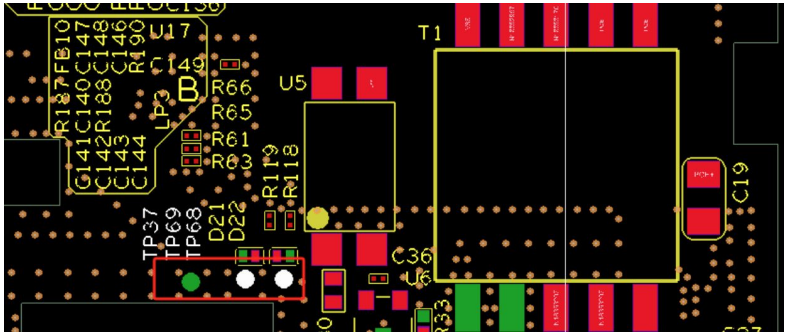
- Availability of dev boards are a good sign
- Supported in your preferred distro (e.g., Yocto)



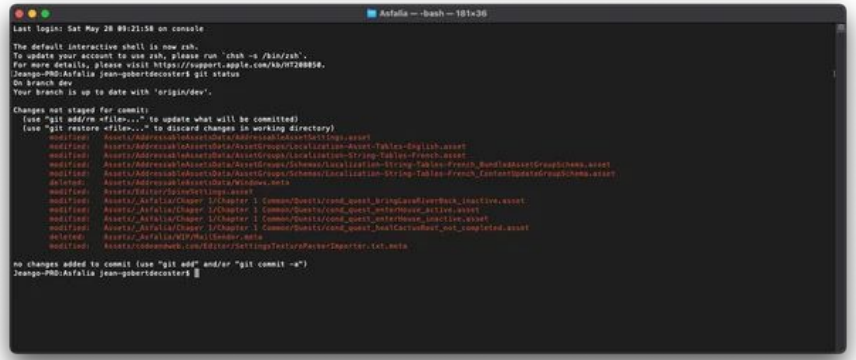



# Key Information To Have

**Schematics** - how peripherals are attached, locations of debug test points at minimum. Full schematics better.




**“SDK”** - This might not be the prettiest. But still useful as a reference.



# Familiarize yourself with the Devicetree

Hardware tends to be actually very similar

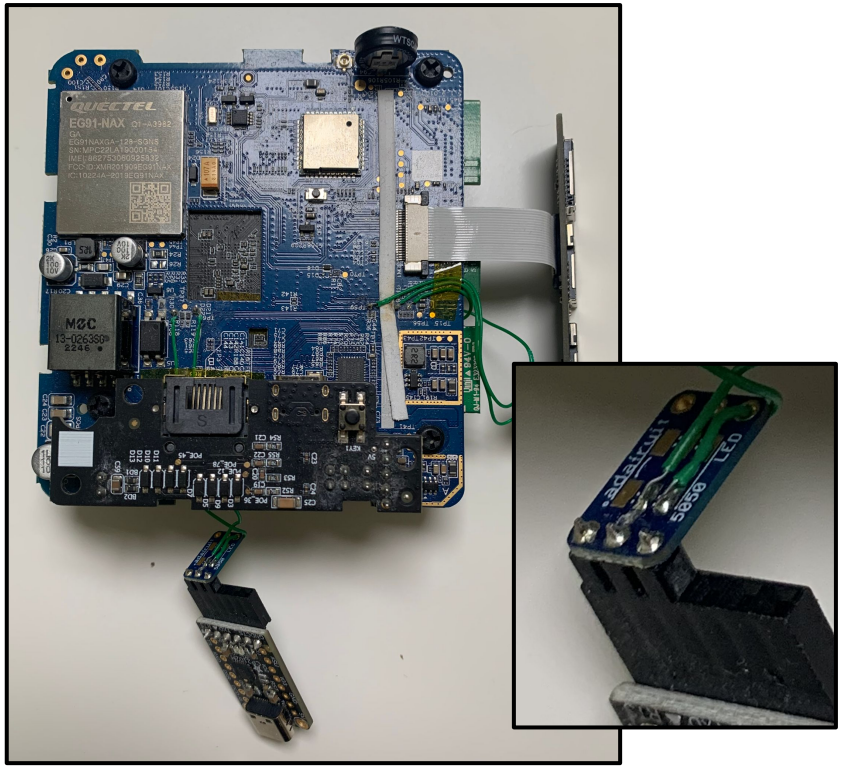
Differences tend to be:

- Which ports are used
- Pinmux settings
- Specific peripherals (LEDs, Wi-Fi / BT, sensors, RTC)

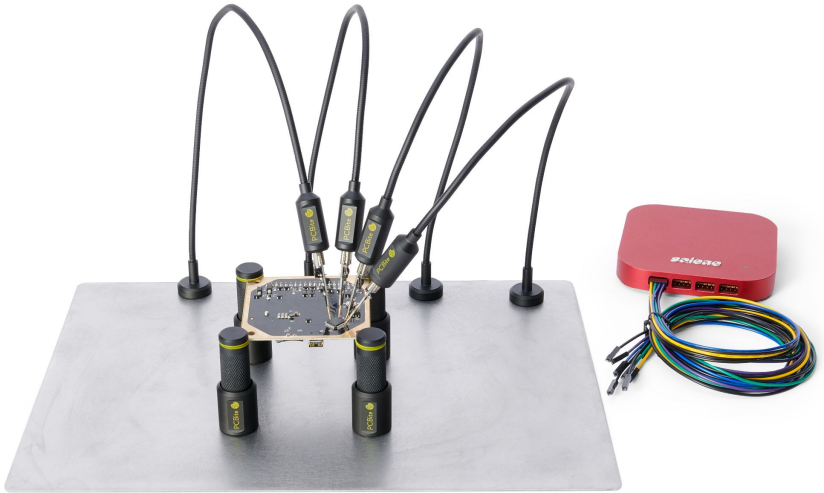
```
&uart1 {
     pinctrl-0 = <&uart1_xfer>;
     status = "okay";
};

&uart2 {
     status = "okay";
};

&uart3 {
     pinctrl-0 = <&uart3m1_xfer>;
     status = "okay";
};
```

# 2. Development

# Developing For Hardware You Didn't Design




Wish I knew about this earlier
(Sensepeek PCBite)



# Experiment with more confidence

Test in emulation

Boot off removable media

Recovery modes on hardware platform

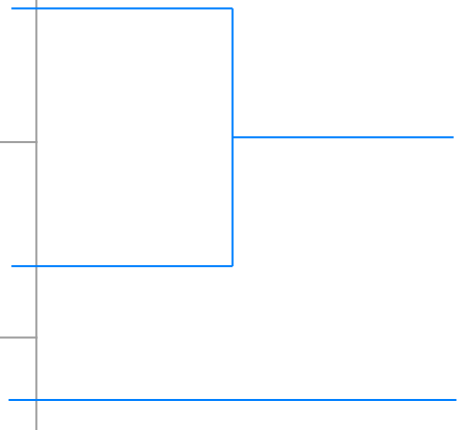
**bootdev** ordering in U-Boot

extlinux.conf

```
LABEL dsgw-093
    KERNEL /boot/fitImage
    APPEND
root=PARTUUID=${uuid}
rootfstype=ext4 rootwait
rw rootfstype=ext4
init=/sbin/preinit
console=ttyS2,1500000n8
```

# Target multiple platforms early on

| Target | Useful For |
| --- | --- |
| Dev board | Known-good reference. Easy access to debug ports and signals |
| Emulator | Easy to reset things that modify the system (e.g., OTA testing) |
| Host PC | Fast iteration loop for applications |



# Yocto Layers

**meta-blecon** hw-independent packages

**meta-blecon-bsp** machine definitions, hw-dependent packages

**Other Layers**

```
rockchip
meta-arm
openembedded
yocto
core
…
```

# 3. Get Ready for Manufacturing

# Integrating into factory's testing flow

Need access to the system to run tools and tests

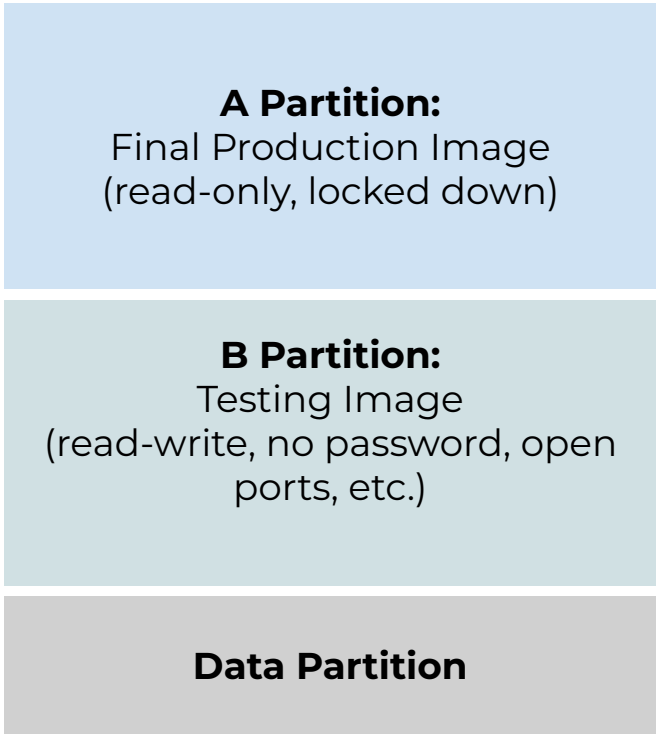
- Radio testing
- Secondary programming
- Manual inspection

Extra rounds of programming prohibitive

# A/B partitions for manufacturing

**Reuse infrastructure for doing OTA Updates**

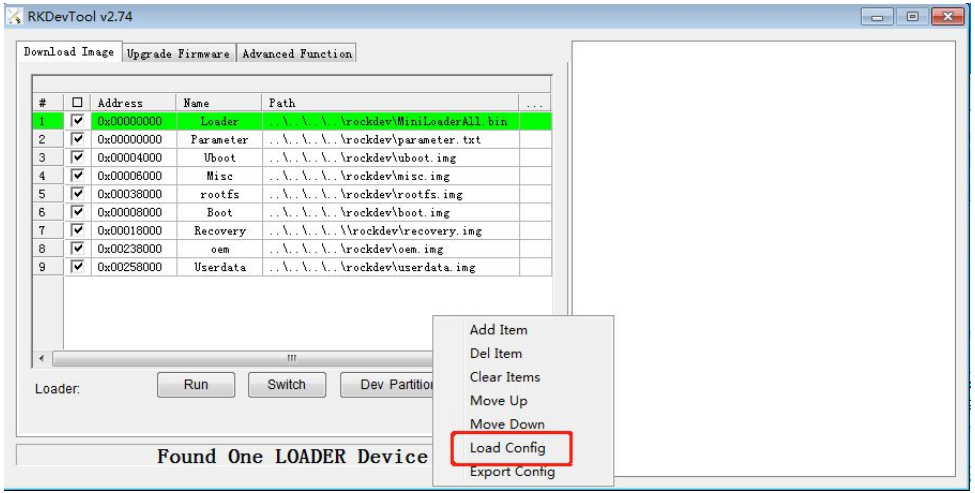
1. QA self-test
2. Write into u-boot environment to flip active partition and reboot
3. U-boot flips bootable flag
4. Boot into prod



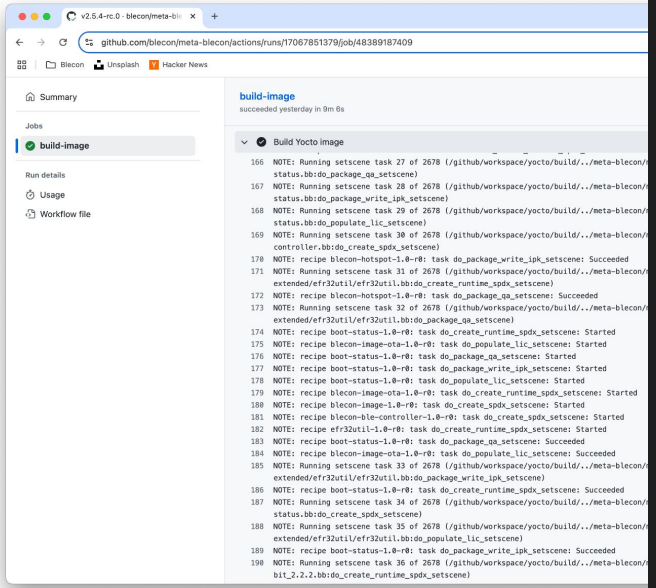
# Manufacturing images != development images

Tools from SoC vendors require special packages (not an SD card image)

- Integrated bootloader
- Description file
- Partition data

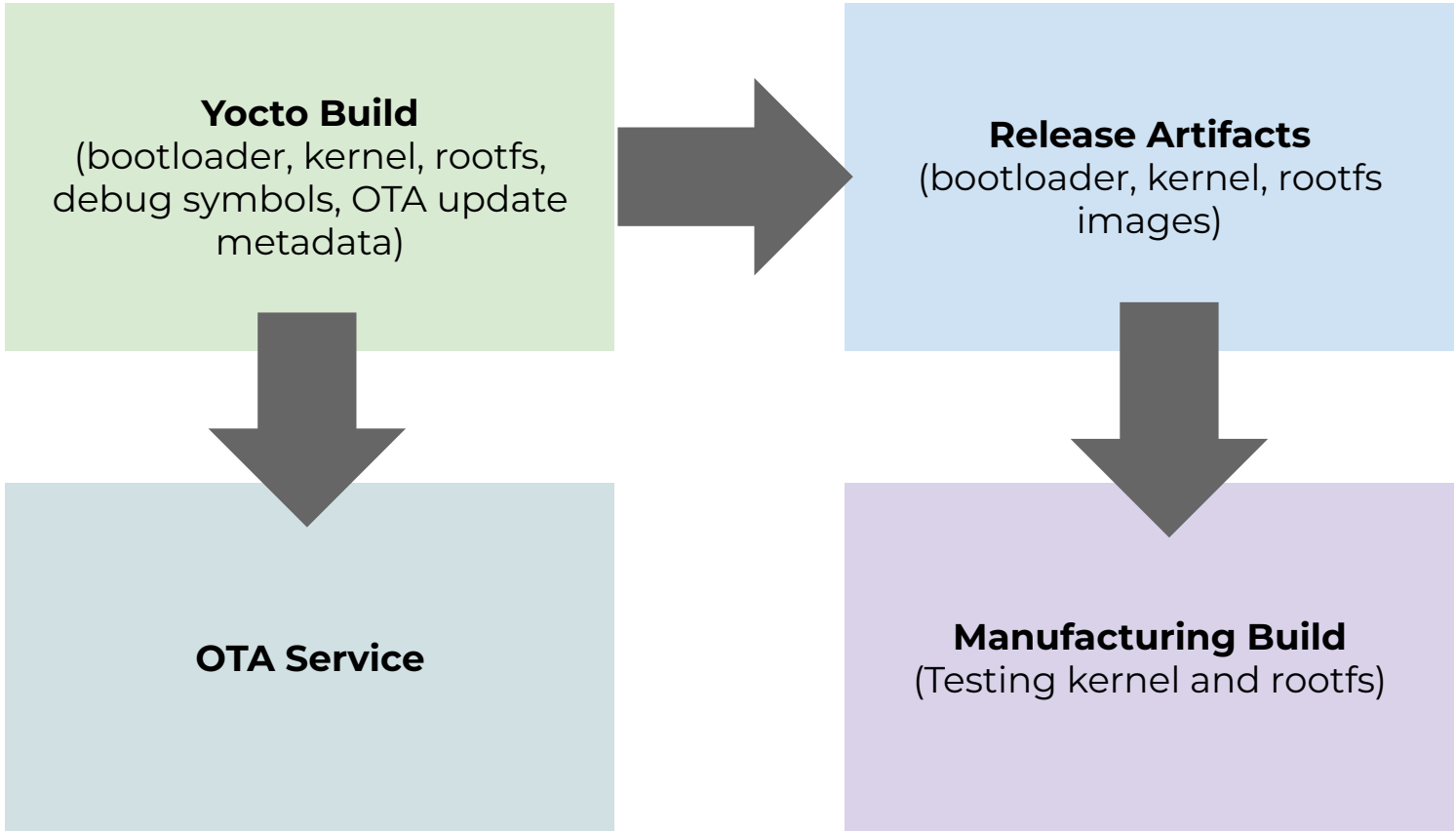


# Build Automation is critical




GitHub Docker Actions

```
west init -l yocto/meta-blecon

cd yocto
west update && west yocto-init --force
chown yocto-build:root build
chmod 775 build

gosu yocto-build git config --global credential.helper

source poky/oe-init-build-env ./build

# Set environment variables to passthrough to Yocto
export BB_ENV_PASSTHROUGH_ADDITIONS="$BB_ENV_PASSTHROUG

gosu yocto-build bitbake $IMAGE_NAME
```


Yocto Build
(bootloader, kernel, rootfs, debug symbols, OTA update metadata)
Release Artifacts
(bootloader, kernel, rootfs images)
OTA Service
Manufacturing Build
(Testing kernel and rootfs)

# And of course, there will be surprises

Differences in testing environment

Unforeseen state

Missed steps

Part changes

**Get golden samples (>1) and a list of things to check**

Cellular modules hide complexity to a dangerous degree

# Now you've received your shipment of devices...

Correlating device IDs / serial numbers

Detecting faults

Managing and flighting OTA

Import duties and HS codes

Cellular weirdness

**Out of time!**

# Conclusion

“Hardware only” white label is an option that benefits on vendors’ experience in hardware and manufacturing

Can result in extremely fast time-to-market, low costs, without sacrificing software quality and security

Some unique challenges around development and manufacturing but open-source packages like Yocto make this *much* easier

# Blecon @ OSS Europe 25

Donatien Garnier

Nick Chen

Adam Dunkels

((o)) Blecon

Zephyr® Project MEMBER

((o)) Blecon

nick.chen@blecon.net

www.linkedin.com/in/nicholas-y-chen