AUTOMOTIVE
LINUX SUMMIT

SOTA Solution
FOTA Solution for AGL

Khiem Nguyen / Renesas
Thu Nguyen / Renesas

@KhiemNguyenT

THE LINUX FOUNDATION

SOTA SOLUTION AND FOTA SOLUTION FOR AGL

JULY 18, 2019
KHIEM NGUYEN - THU NGUYEN
RENESAS DESIGN VIETNAM
RENESAS ELECTRONICS CORPORATION

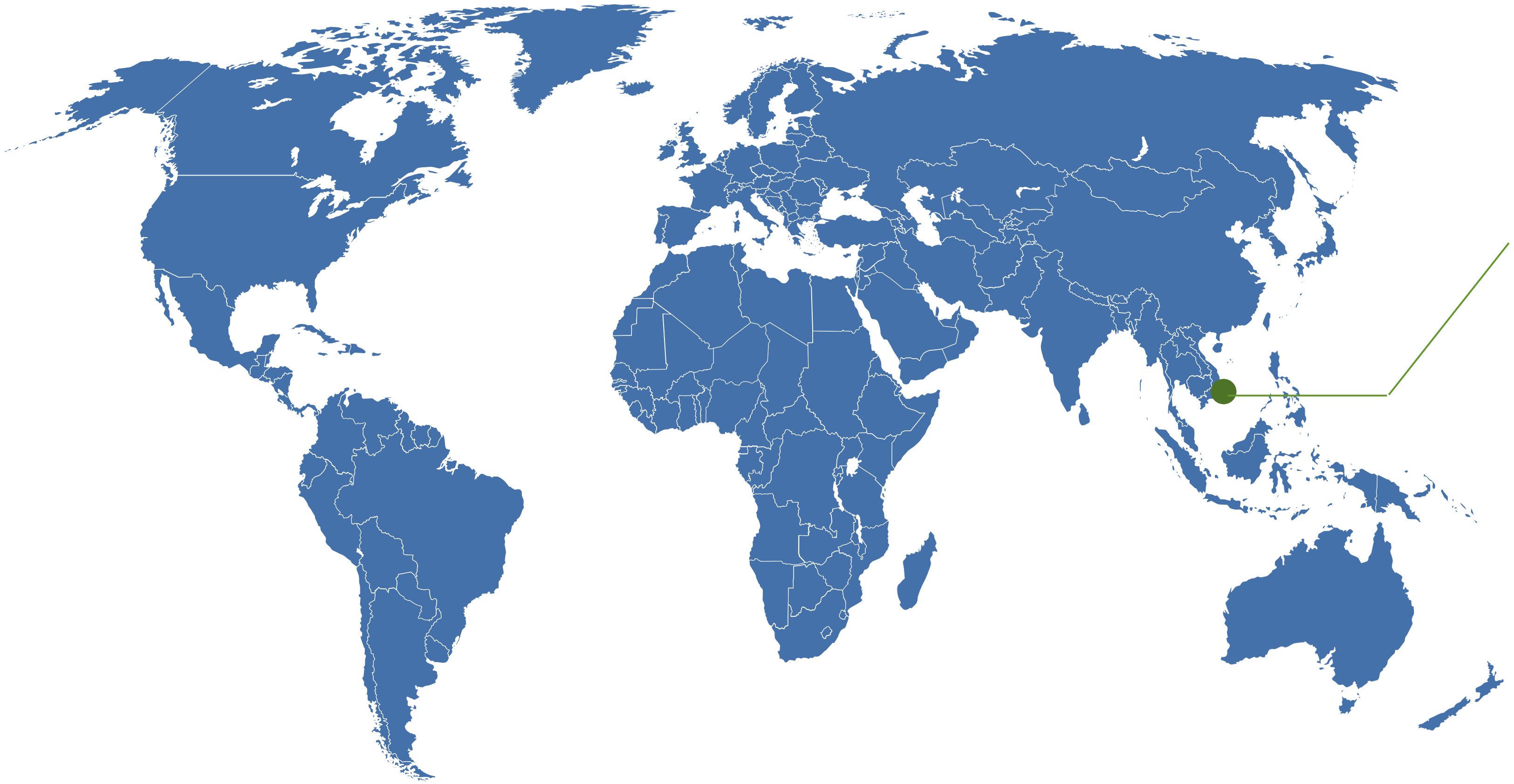
WHO WE ARE ?

- Engineers from Renesas Design Vietnam

- Career:
  - Developer for Mobile and Automotive software platforms.
  - Developer for open-source test automation solutions.
  - Developer for R-Car Gen3 Linux Yocto.

- Email:
  - khiem.nguyen.xt@renesas.com, thu.nguyen.wz@renesas.com

ABOUT RENESAS DESIGN VIETNAM



- Renesas Design Vietnam Co., Ltd. (RVC) was founded in October 2004, as one of the main design centers in Renesas group.
- Business line: Design of semiconductor for both hardware and software.

AGENDA

Motivation	5
Typical OTA requirements and candidate solutions	12
RAUC – Robust Automation Update Controller	13
SOTA and FOTA with RAUC	27
Conclusion & Next plan	35

MOTIVATION

MOTIVATION

WHY WE NEED SOFTWARE UPDATE ?

Fixing issues timely

e.g. Fix CPU Vulnerability, Software incompatibility

Update Security

e.g. Fix CVE of open-source software, Apply LTS update

Support more features

e.g. Annually added features

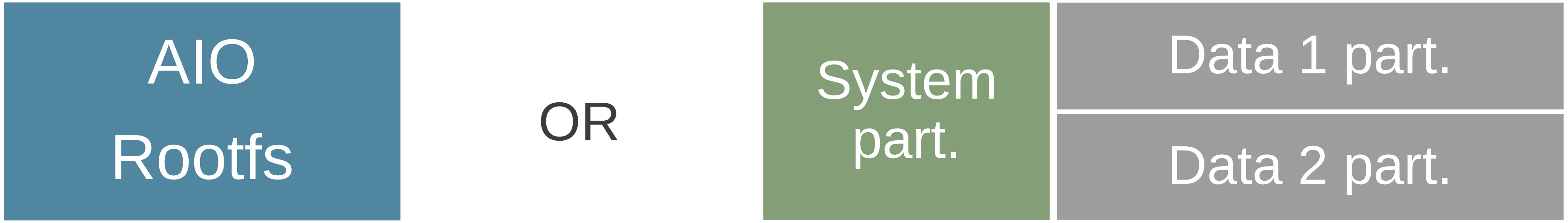
To utilize the high-performance of modern hardware, optimize the system behavior(s) and maintain user satisfaction, software update is demanded feature.

CVE: Common Vulnerabilities and Exposures LTS: Long-term support

MOTIVATION
SOTA AND FOTA

SOTA is **S**oftware **O**ver **T**he **A**ir update.

The software is content of root filesystem which can be managed under one partition or divided into smaller partitions.



AIO Rootfs

OR

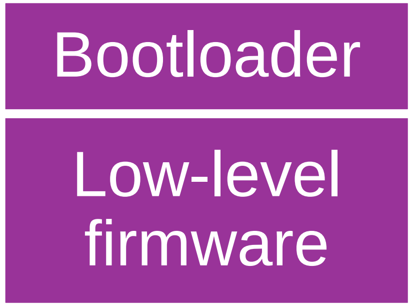
System part.

Data 1 part.

Data 2 part.

FOTA is **F**irmware **O**ver **T**he **A**ir update.

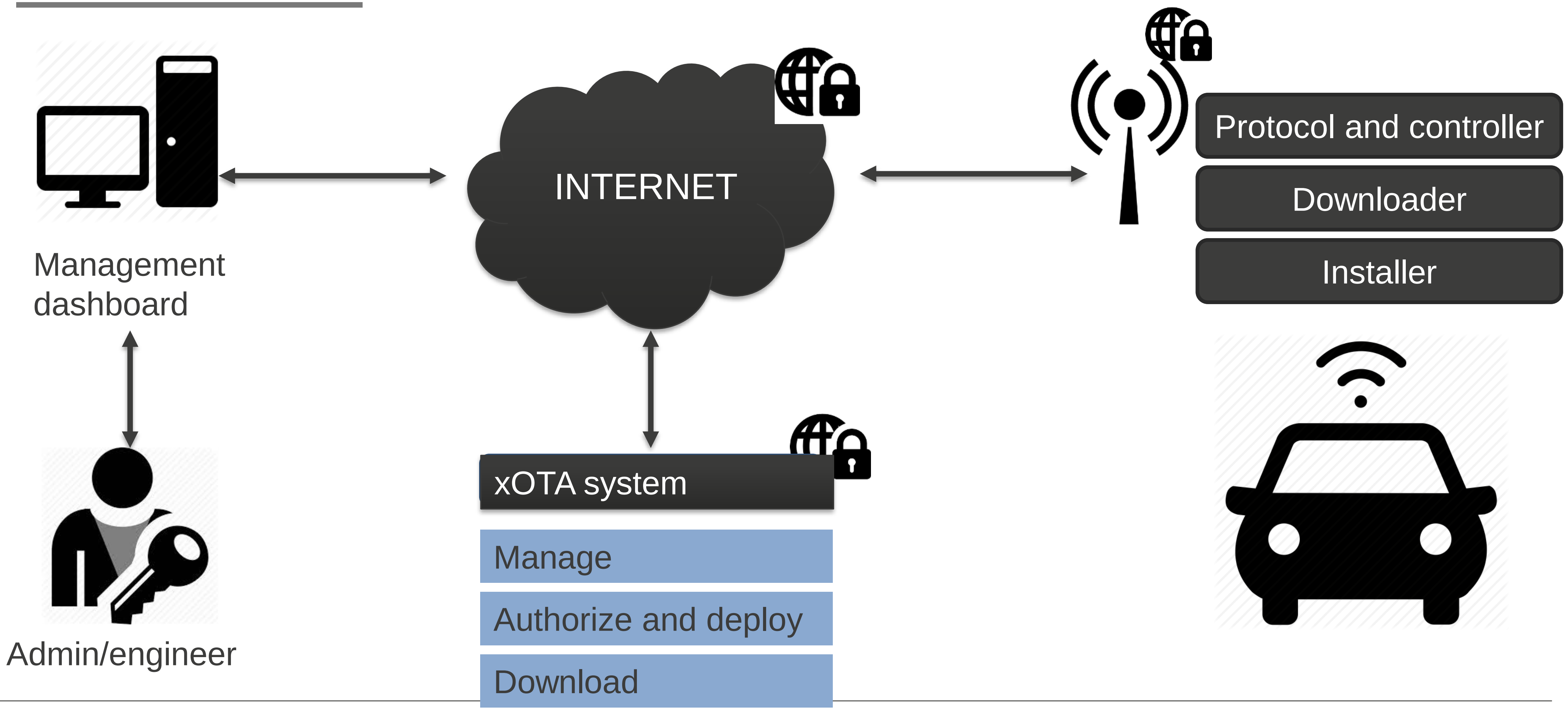
The firmware is the special software which is dedicated for low-level hardware control, secure boot and security services.



Bootloader

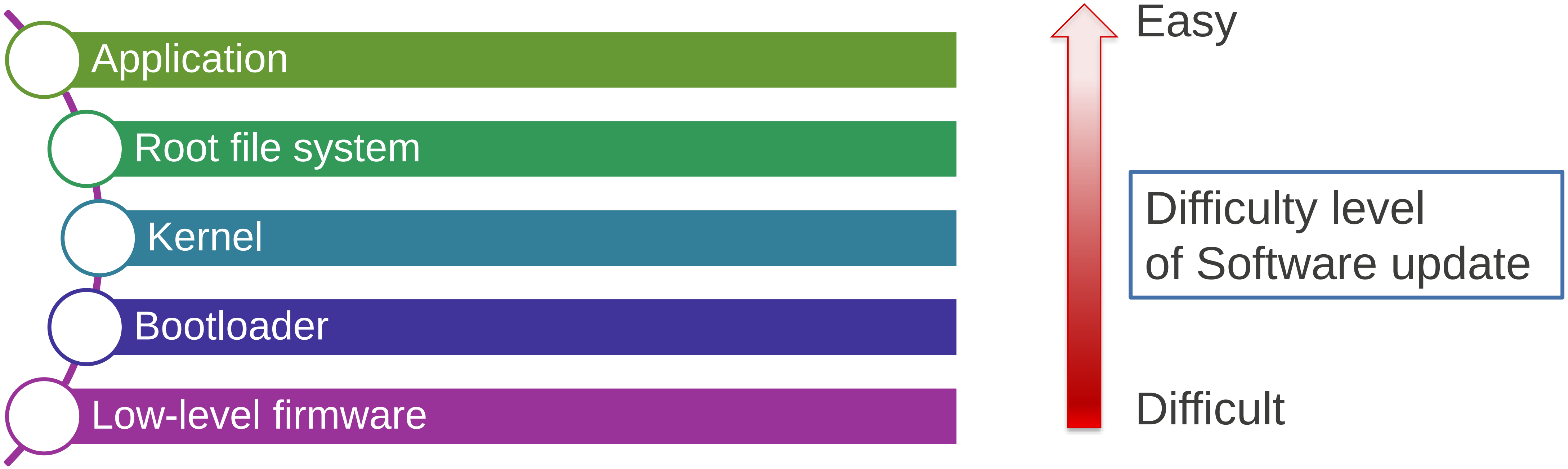
Low-level firmware

MOTIVATION
BASIC OTA ARCHITECTURE



MOTIVATION

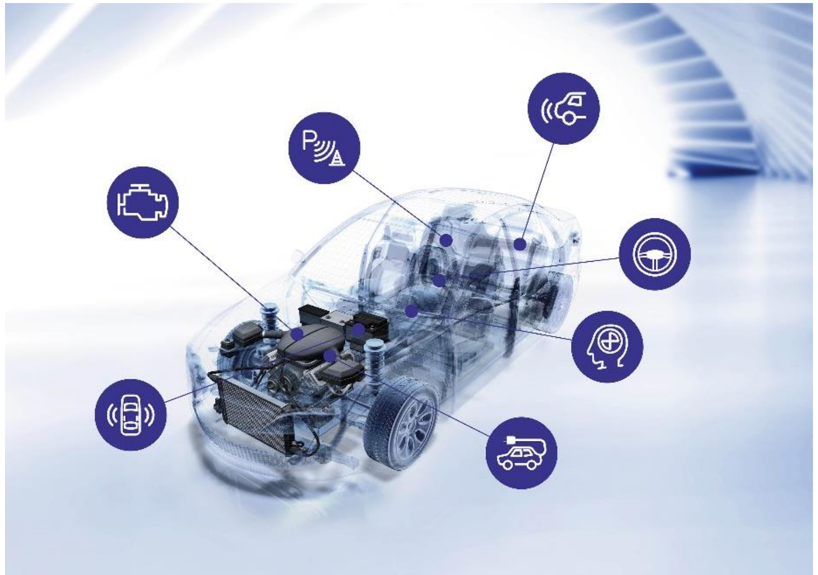
BASIC COMPONENTS OF SW UPDATE IN AUTOMOTIVE SYSTEM



The reference OTA solutions help confirm the system operation for different software update scenarios.

TYPICAL OTA REQUIREMENTS AND CANDIDATE SOLUTIONS

TYPICAL OTA REQUIREMENTS (1/2)
VEHICLE'S SOFTWARE UPDATE EXPECTATION



- Can update the software of automotive device from anywhere.
- Minimizes Security Risks (does not install or execute software created by an attacker).
- Never ends up in an inconsistent state. Keep the device usable (rollback to previous state when there are problems, or at least supporting a recovery mode)
- Requires small additional resources (disk space, RAM).
- Minimizes downtime while updating.

TYPICAL OTA REQUIREMENTS (2/2)
FROM ELC-E 2018 DISCUSSION

- Demanding features for Embedded Software Update solutions (*1):
  - Migration of user data per software update.
  - Alternatives to A/B for constrained systems : support small rescue system.
  - Automatically detection for a successful update.
  - Delta-updates for bandwidth-constrained devices.

(*1) BoF: Embedded Update Tools
https://gist.github.com/jluebbe/d27b2289208791f3805adf69a0dac482

CANDIDATE OTA SOLUTIONS

Tools	RAUC	OSTree	Mender	Swupdate
Update targets	Rootfs, kernel, bootloader	Rootfs and kernel	Rootfs and kernel	Bootloader, kernel, partitions, etc
Update mechanism	Compressed block / file based (tarbal)	File based	Compressed block based	Block / File based
Failure resilience (fallback)	Rollback (needs bootloader support)	Integrated Rollback	Integrated rollback	*No built-in mechanism*
Security	X509-signed update bundles	GPG-signed commits	HTTPS enforced, signed images	HTTPS, signed and encrypted images,

RAUC is a flexible and competent OTA solution for Automotive software.

Reference:
https://wiki.yoctoproject.org/wiki/System_Update

RAUC – ROBUST AUTOMATION UPDATE CONTROLLER

RAUC – ROBUST AUTOMATION UPDATE CONTROLLER

INTRODUCTION (1/3)

- RAUC is an image-based update client. It can update bootloader, kernel, rootfs and applications.
  - The “binary diffs” update is also supported (under development).

- The RAUC update framework provides a solution for four basic tasks:
  - Generate update artifacts
  - Sign and do verification of update artifacts
  - Robust installation handling
  - Interface with the boot process

Reference:
https://rauc.readthedocs.io/en/latest/basic.html

RAUC – ROBUST AUTOMATION UPDATE CONTROLLER

INTRODUCTION (2/3)

- RAUC support some software update scenarios as below:
  - **Symmetric rootfs slots**: A/B partition scheme
  - **Asymmetric Slots**: two slots but the 2nd partition is small, useful for constrained system.
  - **Multiple Slots**: Splitting a system into multiple partitions, useful if the application should be updated independently of the base system. This can be combined with symmetric or asymmetric setups.
  - **Additional Rescue Slot**: adding an additional recovery slot to one of the symmetric scenarios above, when both A and B got trouble during the update.

Reference:
https://rauc.readthedocs.io/en/latest/basic.html

RAUC – ROBUST AUTOMATION UPDATE CONTROLLER

INTRODUCTION (3/3)

- Have Yocto support, provided via meta-rauc layer (*1).
  - For now, it's compatible with Yocto 2.1 to Yocto 2.7.
- Provide RAUC integration example (*2) with Eclipse Hawkbit (*3) deployment server for software rollout operation.
- Support typical bootloaders, i.e. Barebox, U-Boot, GRUB and EFI.

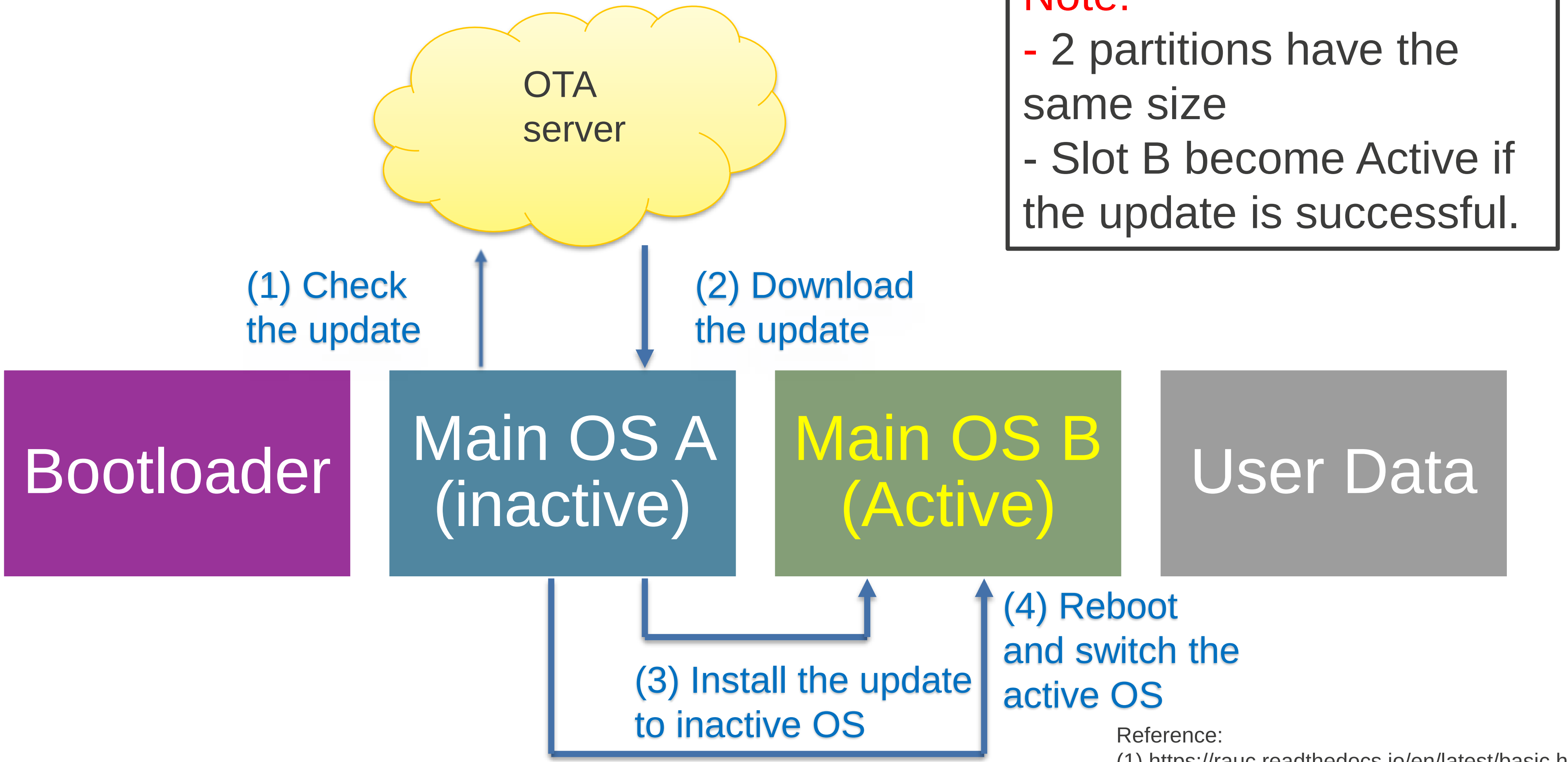
(*1) https://github.com/rauc/meta-rauc
(*2) https://github.com/rauc/rauc-hawkbit
(*3) https://www.eclipse.org/hawkbit/

Reference:
https://rauc.readthedocs.io/en/latest/basic.html

RAUC – SYMMETRIC UPDATE

OVERVIEW

Note:
- 2 partitions have the same size
- Slot B become Active if the update is successful.



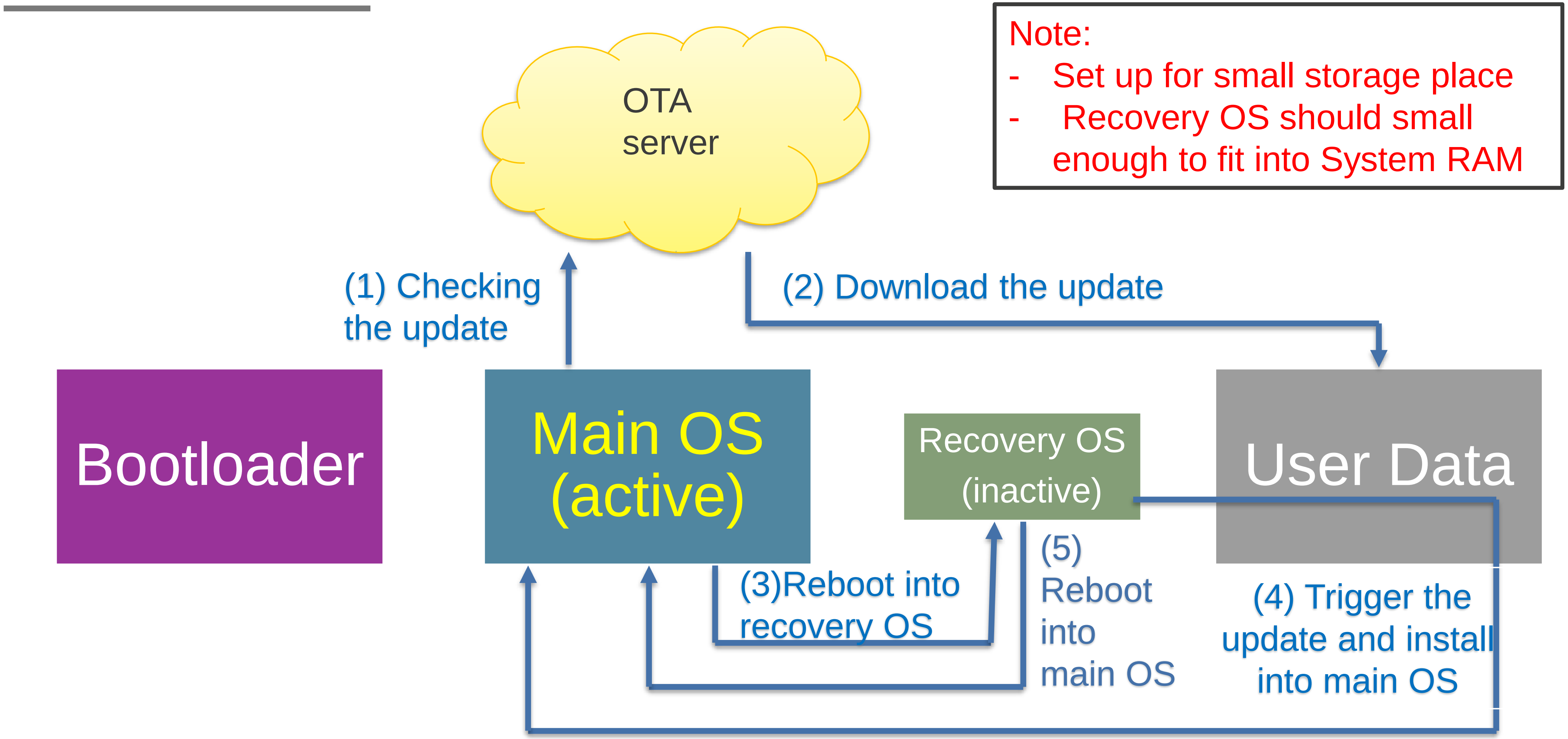
Reference:
(1) https://rauc.readthedocs.io/en/latest/basic.html

RAUC – ASYMMETRIC UPDATE

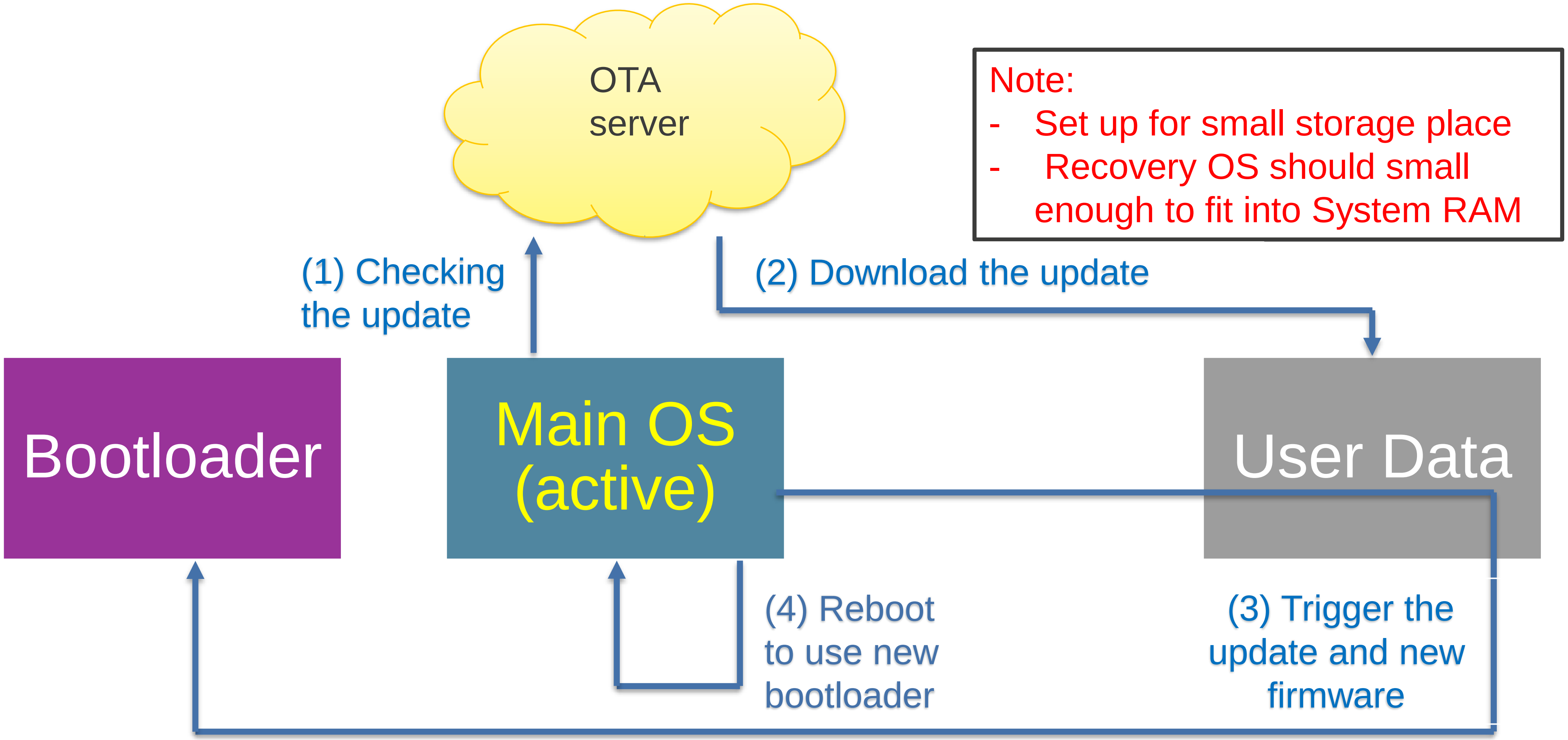
OVERVIEW

Note:

- Set up for small storage place
- Recovery OS should small enough to fit into System RAM



RAUC – FIRMWARE (BOOTLOADER) UPDATE



Note:
- Set up for small storage place
- Recovery OS should small enough to fit into System RAM

SOTA AND FOTA WITH RAUC

SOTA AND FOTA WITH RAUC

YOCTO RECIPE INTEGRATION

1. Install rauc into AGL environment:

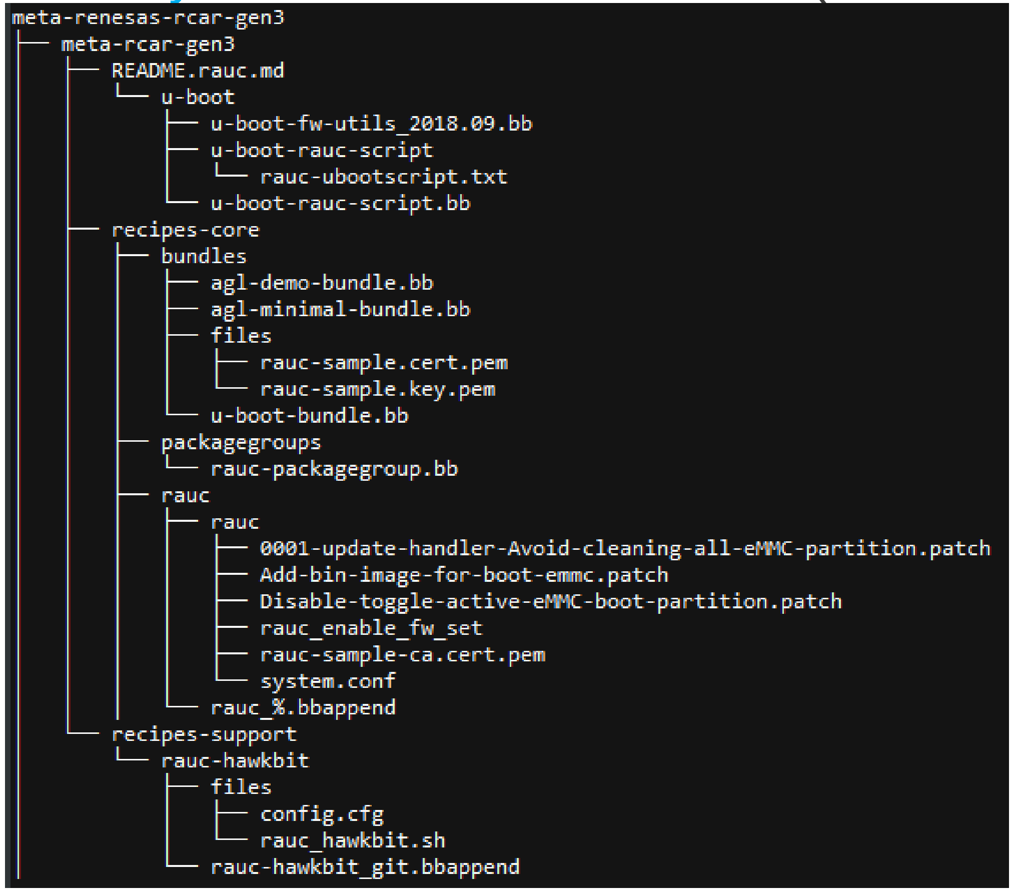
- In local.conf, add config as below:

```
IMAGE_INSTALL_append = " rauc"
```

- In bblayer.conf add line as below:

```
BBLAYERS =+ " \
   ${METADIR}/meta-renesas-rcar-gen3 \
   ${METADIR}/meta-agl/meta-agl-bsp \
   ${METADIR}/meta-rauc \
   "
```

Meta layer to install RAUC to AGL (for m3ulcb)



```
meta-renesas-rcar-gen3
├── meta-rcar-gen3
│   ├── README.rauc.md
│   │   └── u-boot
│   │       ├── u-boot-fw-utils_2018.09.bb
│   │       ├── u-boot-rauc-script
│   │       │   └── rauc-ubootscript.txt
│   │       └── u-boot-rauc-script.bb
│   ├── recipes-core
│   │   ├── bundles
│   │   │   ├── agl-demo-bundle.bb
│   │   │   ├── agl-minimal-bundle.bb
│   │   │   ├── files
│   │   │   │   ├── rauc-sample.cert.pem
│   │   │   │   └── rauc-sample.key.pem
│   │   │   └── u-boot-bundle.bb
│   │   ├── packagegroups
│   │   │   └── rauc-packagegroup.bb
│   │   ├── rauc
│   │   │   ├── rauc
│   │   │   │   ├── 0001-update-handler-Avoid-cleaning-all-eMMC-partition.patch
│   │   │   │   ├── Add-bin-image-for-boot-emmc.patch
│   │   │   │   ├── Disable-toggle-active-eMMC-boot-partition.patch
│   │   │   │   ├── rauc_enable_fw_set
│   │   │   │   ├── rauc-sample-ca.cert.pem
│   │   │   │   └── system.conf
│   │   │   └── rauc_%.bbappend
│   └── recipes-support
│       └── rauc-hawkbit
│           ├── files
│           │   ├── config.cfg
│           │   └── rauc_hawkbit.sh
│           └── rauc-hawkbit_git.bbappend
│
```

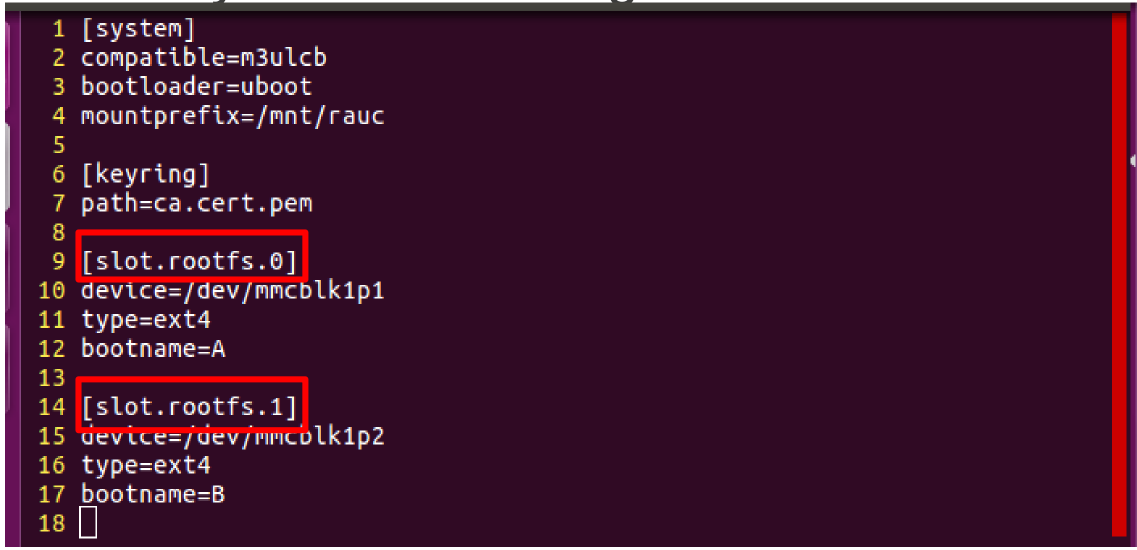
SOTA AND FOTA WITH RAUC

RAUC SYSTEM CONFIGURATION

2. RAUC configuration and setting :

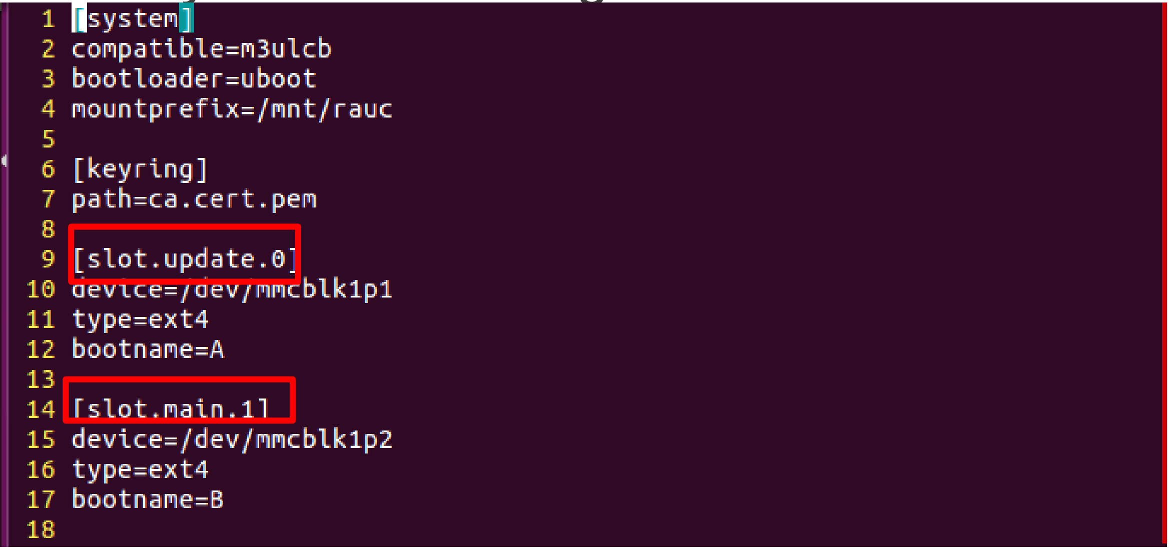
- Generate keyring, key, certification: refer script in (*1)
- Config for rauc:

+ Symmetric setting:



```
[system]
compatible=m3ulcb
bootloader=uboot
mountprefix=/mnt/rauc

[keyring]
path=ca.cert.pem

[slot.rootfs.0]
device=/dev/mmcblk1p1
type=ext4
bootname=A

[slot.rootfs.1]
device=/dev/mmcblk1p2
type=ext4
bootname=B

```

+ Asymmetric setting:



```
[system]
compatible=m3ulcb
bootloader=uboot
mountprefix=/mnt/rauc

[keyring]
path=ca.cert.pem

[slot.update.0]
device=/dev/mmcblk1p1
type=ext4
bootname=A

[slot.main.1]
device=/dev/mmcblk1p2
type=ext4
bootname=B

```

(*1) https://github.com/rauc/meta-rauc/tree/master/scripts

SOTA AND FOTA WITH RAUC

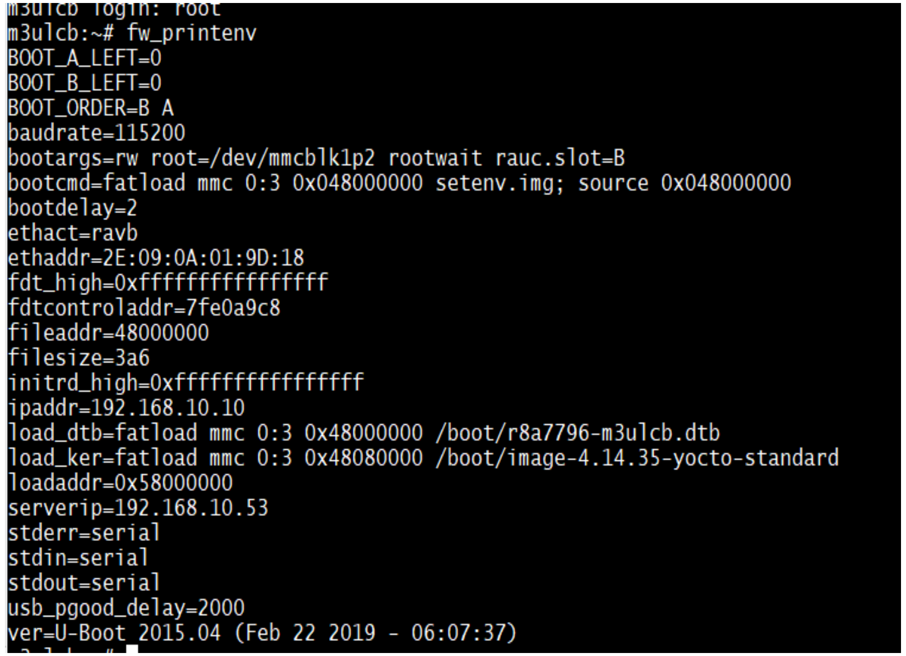
ADDITIONAL SETTING IN BOOTLOADER

2. Notice in configuration and setting :

- Config for auto switch OS (U-boot seting):
  - Install uboot-fw-utils package

    IMAGE_INSTALL_append = " uboot-fw-utils"

  - When build successfully, we will have fw_printenv, fw_setenv in rootfs.
- Register device node name which is store U-boot environment variables to /etc/fw_env.config.
- Create a script which is help U-boot choosing bootargs automatically(*1).
- Use mkimage to convert U-boot script file to a script image.

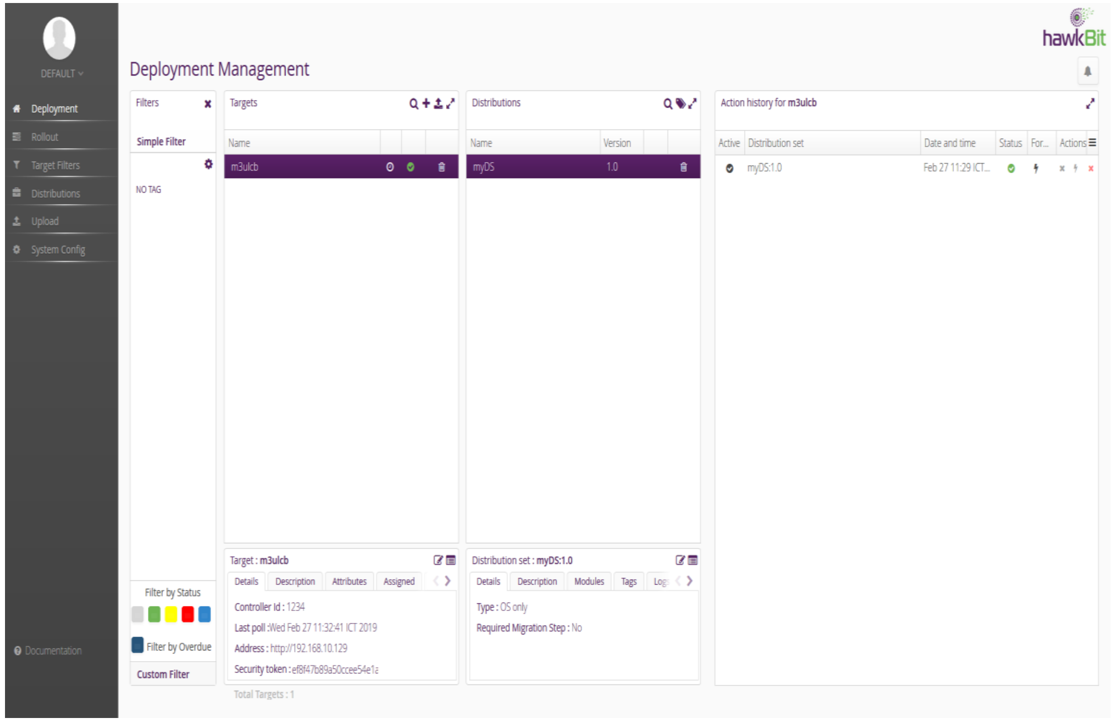


```
m3ulcb login: root
m3ulcb:~# fw_printenv
BOOT_A_LEFT=0
BOOT_B_LEFT=0
BOOT_ORDER=B A
baudrate=115200
bootargs=rw root=/dev/mmcblk1p2 rootwait rauc.slot=B
bootcmd=fatload mmc 0:3 0x048000000 setenv.img; source 0x048000000
bootdelay=2
ethact=ravb
ethaddr=2E:09:0A:01:9D:18
fdt_high=0xffffffffffffffff
fdtcontroladdr=7fe0a9c8
fileaddr=48000000
filesize=3a6
initrd_high=0xffffffffffffffff
ipaddr=192.168.10.10
load_dtb=fatload mmc 0:3 0x48000000 /boot/r8a7796-m3ulcb.dtb
load_ker=fatload mmc 0:3 0x48080000 /boot/image-4.14.35-yocto-standard
loadaddr=0x58000000
serverip=192.168.10.53
stderr=serial
stdin=serial
stdout=serial
usb_pgood_delay=2000
ver=U-Boot 2015.04 (Feb 22 2019 - 06:07:37)
```

(*1)https://github.com/rauc/rauc/blob/master/contrib/uboot.sh

SOTA AND FOTA WITH RAUC

DEPLOY HAWKBIT FOR SOFTWARE ROLEOUT



Deployment Management

Targets

Name
m3ulcb

Distributions

Name	Version
myDS	1.0

Action history for m3ulcb

Active	Distribution set	Date and time	Status	For...	Actions
✓	myDS:1.0	Feb 27 11:29 ICT...	✓		

Target : m3ulcb

Controller Id : 1234

Last poll :Wed Feb 27 11:32:41 ICT 2019

Address : http://192.168.10.129

Security token : ef8f47b89a50ccee54e1a

Total Targets : 1

Distribution set : myDS:1.0

Type : OS only

Required Migration Step : No

3. Setup SOTA (hawkbit) server/client

- For hawkbit client: on target system
  - In local.conf, add config as below:

```
IMAGE_INSTALL_append = " rauc-hawkbit“
```

  - After that, rebuild the rootfs system.
    - If build successfully, there’s /usr/lib/rauc-hawkbit-client in new rootfs.
- For hawkbit server: on Host PC
  - Please refer to (*1) to install and start hawkbit server from docker image.
  - After start successfully, the hawkbit server GUI can be accessed.

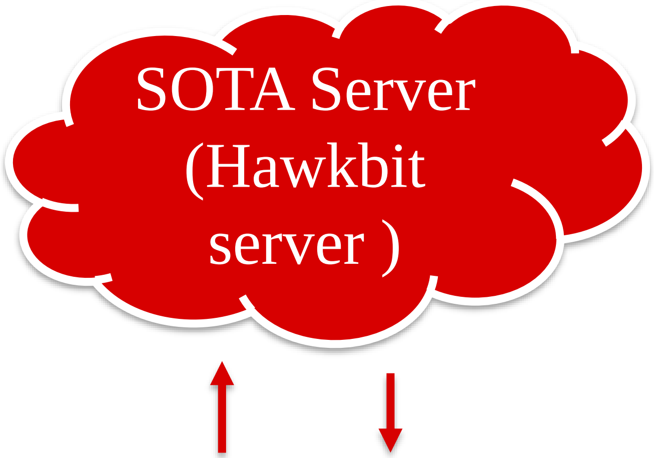
(*1) https://www.eclipse.org/hawkbit/gettingstarted/#from-docker-image

SOTA AND FOTA WITH RAUC

DEMO

Demo 1: Rootfs update

(Symmetric scenario)



SOTA Server (Hawkbit server)

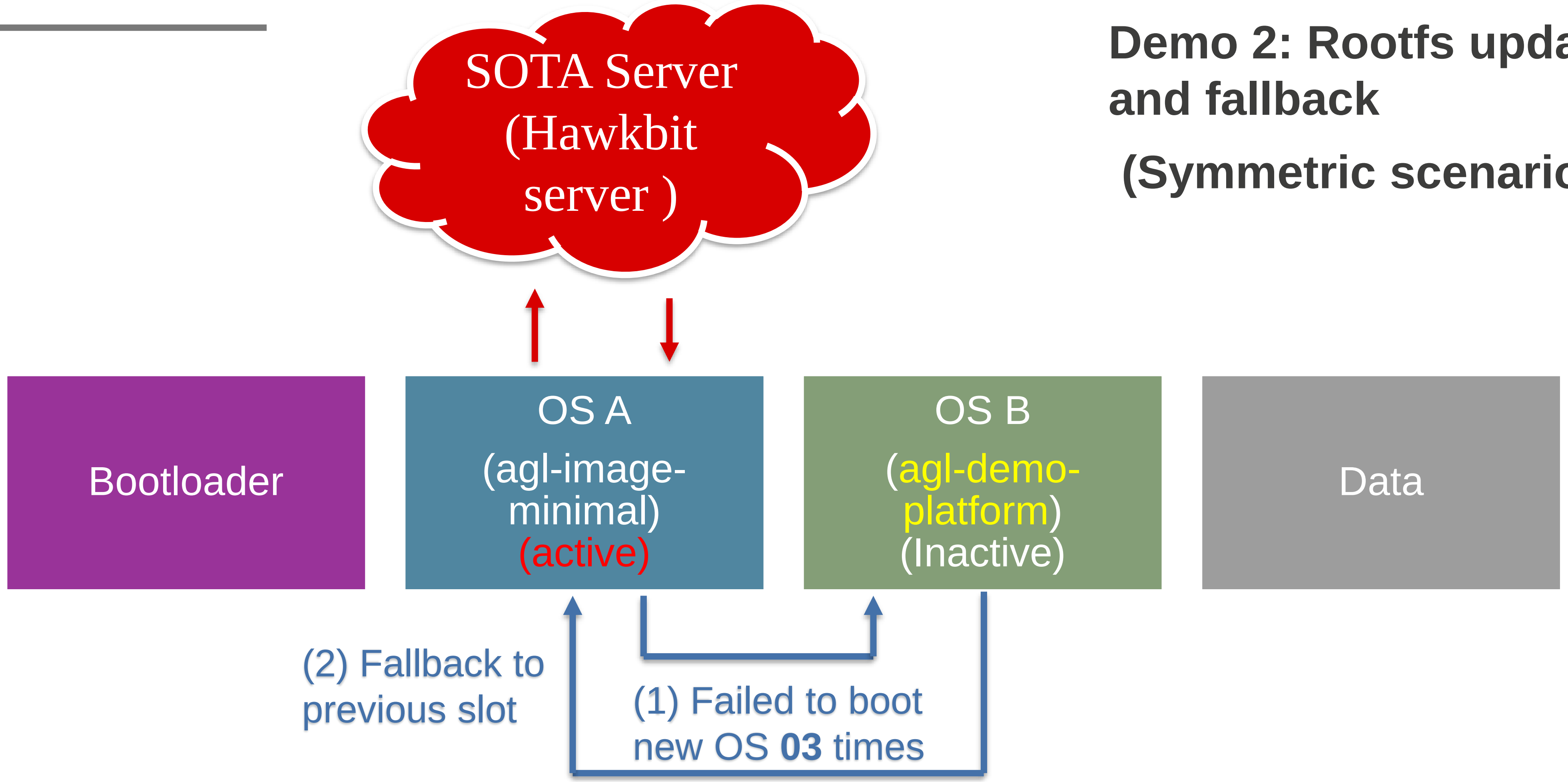
Before	Bootloader	OS A (agl-image-minimal) (active)	OS B (agl-image-minimal) (Inactive)	Data
After	Bootloader	OS A (agl-image-minimal) (Inactive)	OS B (agl-demo-platform) (Active)	Data

SOTA AND FOTA WITH RAUC

DEMO

Demo 2: Rootfs update and fallback

(Symmetric scenario)

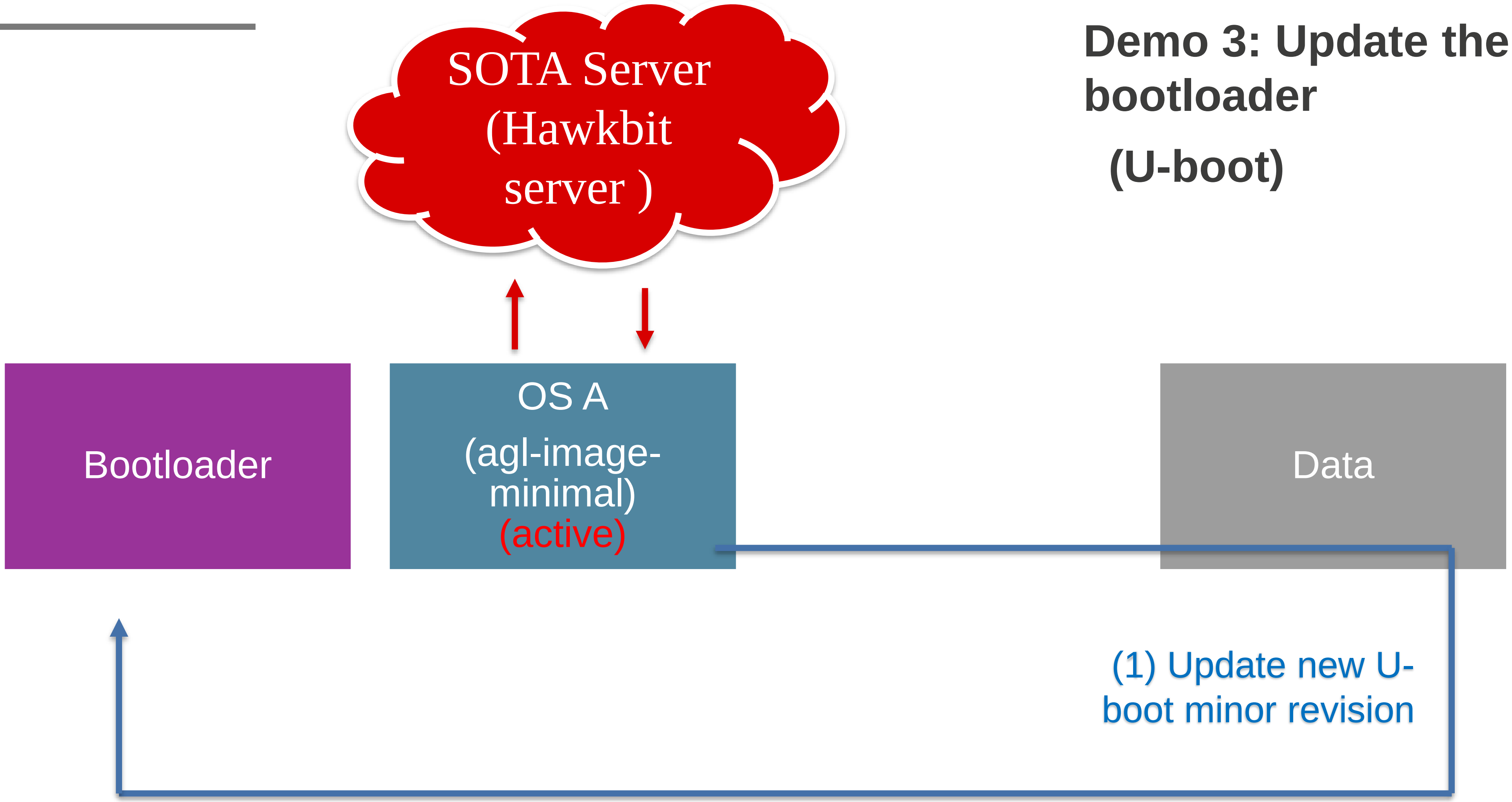


SOTA AND FOTA WITH RAUC

DEMO

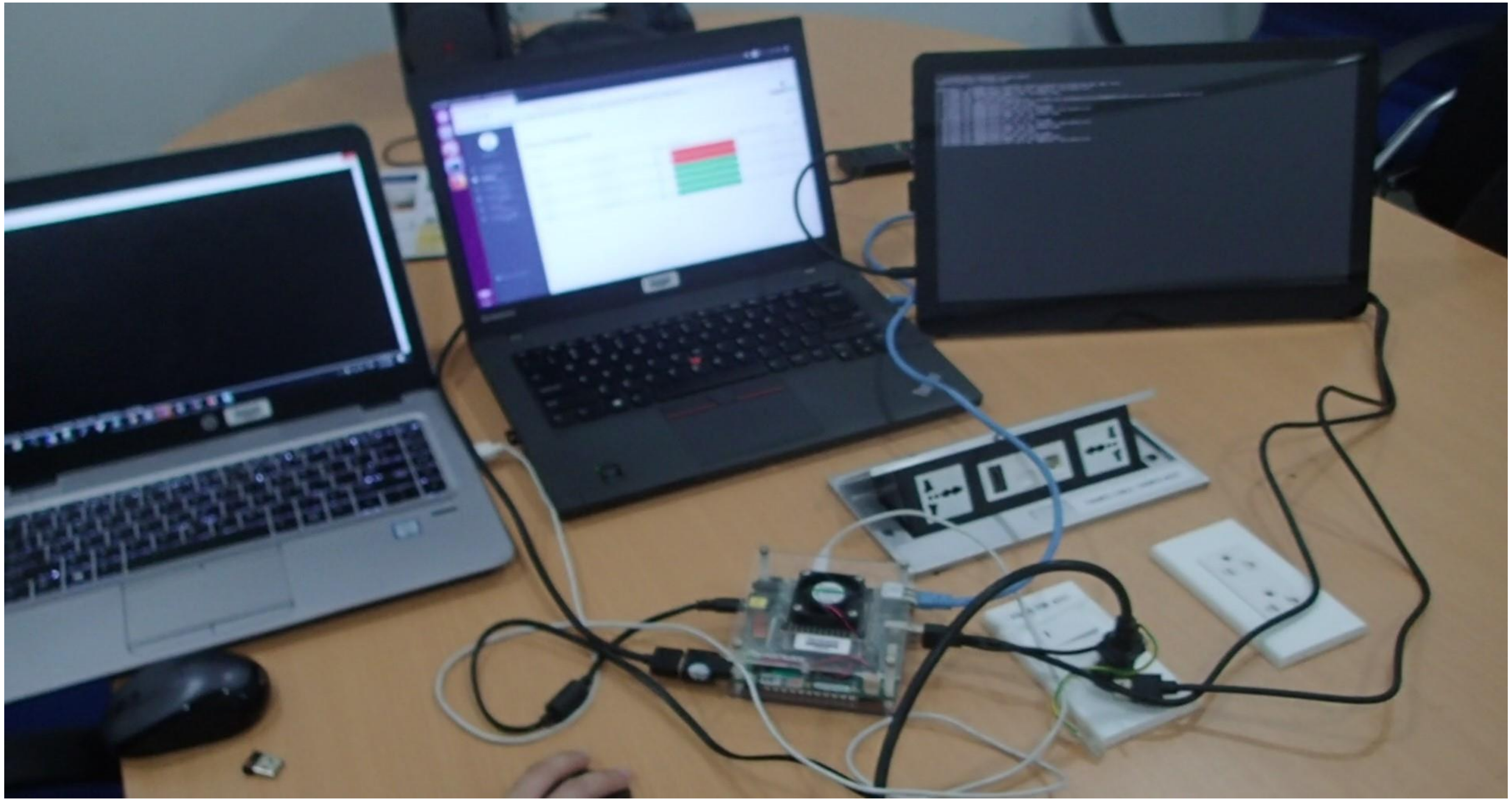
Demo 3: Update the bootloader

(U-boot)


SOTA Server (Hawkbit server)
Bootloader
OS A
(agl-image-minimal)
(active)
Data
(1) Update new U-boot minor revision

SOTA AND FOTA WITH RAUC

DEMO



CONCLUSION AND NEXT PLAN

CONCLUSION AND NEXT PLAN

- Software update (SOTA and FOTA) is an important and demanding technology in Automotive industry.
  - RAUC is a software update solution which is flexible, Yocto-compatible and easy to use for AGL distribution.

- Next plan
  - Share the Yocto recipe to support RAUC with R-Car M3 Starter Kit.
  - Consider solution for low-level firmware update.
  - Consider fallback solution for firmware update.

Renesas.com

Q&A

AUTOMOTIVE LINUX SUMMIT