SOTA Solution
FOTA Solution for AGL

Khiem Nguyen / Renesas
Thu Nguyen / Renesas

@KhiemNguyenT
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
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 *Software Over The Air* update.

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

**FOTA** is *Firmware Over The Air* update.

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

INTERNET

Protocol and controller
Downloader
Installer

Management dashboard

Admin/engineer

xOTA system
Manage
Authorize and deploy
Download

Protocol and controller
Downloader
Installer

© 2019 Renesas Electronics Corporation. All rights reserved.
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

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

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

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

© 2019 Renesas Electronics Corporation. All rights reserved.
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

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 2\textsuperscript{nd} 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.

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.

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

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

**OVERVIEW**

![Diagram of RAUC symmetric update process](image)

1. **Check the update**
2. **Download the update**
3. **Install the update to inactive OS**
4. **Reboot and switch the active OS**

**Main OS A (inactive)**

**Main OS B (Active)**

**User Data**

**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 be small enough to fit into System RAM

(1) Checking the update
(2) Download the update
(3) Reboot into recovery OS
(4) Trigger the update and install into main OS
(5) Reboot into main OS

OTA server

Bootloader

Main OS (active)

Recovery OS (inactive)

User Data

OTA server
RAUC – FIRMWARE (BOOTLOADER) UPDATE

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

1. Checking the update
2. Download the update
3. Trigger the update and new firmware
4. Reboot to use new bootloader
SOTA AND FOTA WITH RAUC
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 += "\n     ${METADIR}/meta-renesas-rcar-gen3 \n     ${METADIR}/meta-agl/meta-agl-bsp \n     ${METADIR}/meta-rauc \n     "
     ```
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
  [slot_rootfs.1]
  device=/dev/mmcblk1p2
  type=ext4
  ```

  + Asymmetric setting:

  ```
  [system]
  compatible=m3ulcb
  bootloader=uboot
  mountprefix=/mnt/rauc
  [keyring]
  path=ca.cert.pem
  [slot_update.0]
  device=/dev/mmcblk1p1
  type=ext4
  [slot_main.1]
  device=/dev/mmcblk1p2
  type=ext4
  ```

(*1) https://github.com/rauc/meta-rauc/tree/master/scripts
2. Notice in configuration and setting:

- Config for auto switch OS (U-boot seting):
  - Install u-boot-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.

SOTA AND FOTA WITH RAUC
DEPLOY HAWKBIT FOR SOFTWARE ROLEOUT

3. Setup SOTA (hawkbit) server/client

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

    ```plaintext
    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)

Before

<table>
<thead>
<tr>
<th>Bootloader</th>
<th>OS A (agl-image-minimal) (active)</th>
<th>OS B (agl-image-minimal) (Inactive)</th>
<th>Data</th>
</tr>
</thead>
</table>

After

<table>
<thead>
<tr>
<th>Bootloader</th>
<th>OS A (agl-image-minimal) (Inactive)</th>
<th>OS B (agl-demo-platform) (Active)</th>
<th>Data</th>
</tr>
</thead>
</table>
**SOTA AND FOTA WITH RAUC DEMO**

![Diagram with SOTA Server (Hawkbit server) and OS A (agl-image-minimal) (active) and OS B (agl-demo-platform) (Inactive) with arrows indicating (1) Failed to boot new OS 03 times and (2) Fallback to previous slot]

Demo 2: Rootfs update and fallback (Symmetric scenario)

- **Bootloader**
- **OS A** (agl-image-minimal) (active)
- **OS B** (agl-demo-platform) (Inactive)

© 2019 Renesas Electronics Corporation. All rights reserved.
Demo 3: Update the bootloader (U-boot)

(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.
Q&A