



## Linux Embarqué

Mises à jour logicielles en environnement Linux Embarqué,  
petit guide et tour d'horizon

---

Pierre-Jean Texier

Samedi 17 Novembre

Toulouse, CDL 2018

/me

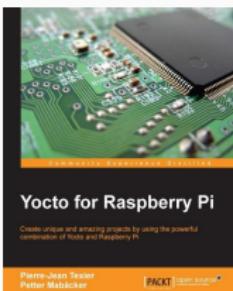
---

# Pierre-Jean Texier

- Ingénieur Linux Embarqué **Lafon Technologies**



- 28 ans
- FOSS enthusiast
- Contributions : U-Boot, Kernel Linux, Yocto/OE, ...
- Co-auteur de "**Yocto for Raspberry Pi**" et auteur dans GNU/Linux magazine et Open silicium (RIP)



# Sommaire

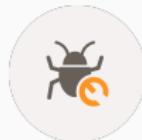
1. Mise à jour des systèmes embarqués
2. Les différents projets Open-Source
3. RAUC, WaRP7 & Yocto/OE
4. Mender, WaRP7 & Yocto/OE
5. Démos
6. Conclusion

# **Mise à jour des systèmes embarqués**

---

# Pourquoi un système de mise à jour ?

- Bugs Logiciels



- Ajouts de fonctionnalités



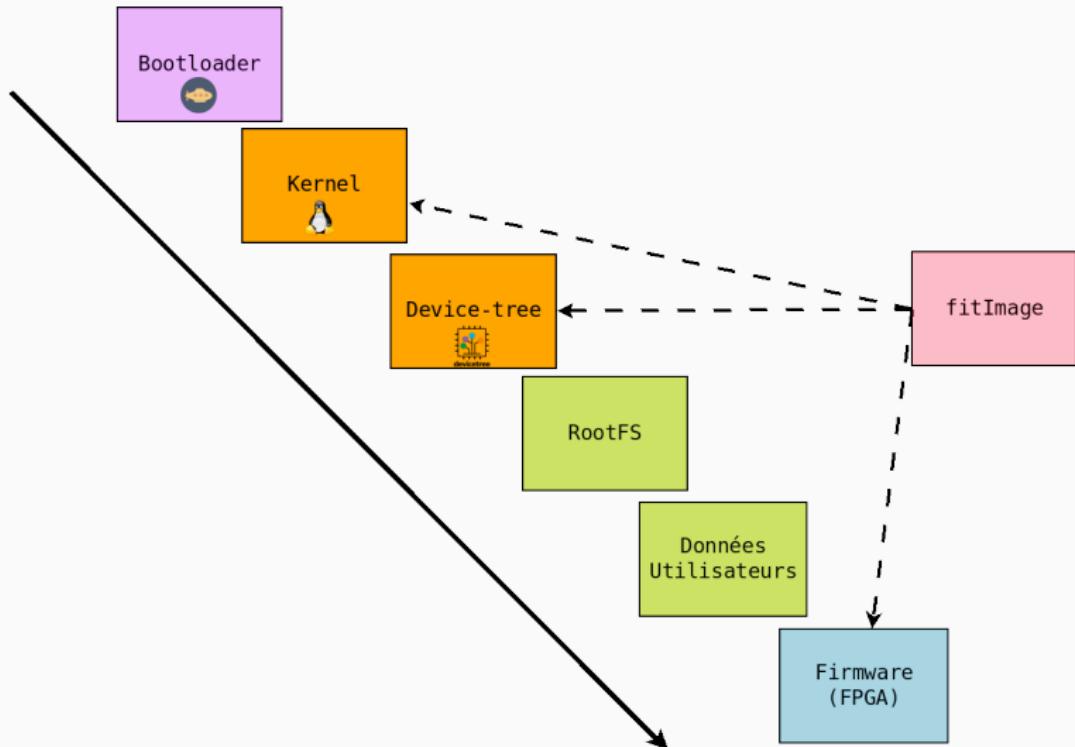
- Correctifs liés à la sécurité (CVE)



# Pourquoi si spécial dans l'embarqué ?

- Accessibilité
- Disponibilité
- Fiabilité du réseau électrique
  - Système sur batterie,
  - Perte de tension (secteur)
- Connectivités réseaux (pour les mises à jour distantes)
  - GPRS, 3G, LTE, WiFi, ...
    - Faible bande passante
- Microcontrôleur, Microprocesseur, FPGA, ...
- Durée de vie sur site supérieure à 10 ans

# Ok, mais quels composants ?



# fitImage

- zImage

```
/dts-v1/;

{
    description = "CDL fitImage";
    #address-cells = <1>;

    images {
        kernel {
            description = "zImage";
            data = /incbin("./zImage");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "none";
            load = <0x80800000>;
            entry = <0x80800000>;
            hash-1 {
                algo = "sha256";
            };
        };
    ...
}
```

# fitImage

- FDT

```
fdt-1 {  
    description = "Flattened Device Tree blob";  
    data = /incbin/("./imx7s-warp.dtb");  
    type = "flat_dt";  
    arch = "arm";  
    compression = "none";  
    ...  
};  
};
```

- Firmware

```
firmware-1 {  
    description = "Firmware Cortex M4";  
    data = /incbin/("./main.bin");  
    type = "firmware";  
    arch = "arm";  
    compression = "none";  
    ...  
};
```

# fitImage

- Configuration

```
configurations {
    default = "conf-1";
    conf-1 {
        description = "CDL Configuration";
        kernel = "kernel";
        fdt = "fdt-1";
        firmware = "firmware-1";
        signature {
            algo = "sha256,rsa4096";
            key-name-hint = "cdl-key";
            sign-images = "firmware", "fdt", "kernel";
        };
    };
};
```

- Chargement

```
=> load mmc 0:1 0x85000000 fitImage
=> bootm 0x85000000
```

# Sur quelle base ?

- Fichiers
  - A éviter, difficile de garantir l'atomicité (-)

# Sur quelle base ?

- **Fichiers**
  - A éviter, difficile de garantir l'atomicité (-)
- **Gestionnaire de paquets**
  - RPM, deb, opkg
  - `rpm -ivh my-wonderful-app.rpm` ? (-)
  - Facile (+) mais difficile à maintenir -> gestion des dépendances (-)
  - Non Atomique et non applicable pour l'embarqué (-)

# Sur quelle base ?

- **Fichiers**
  - A éviter, difficile de garantir l'atomicité (-)
- **Gestionnaire de paquets**
  - RPM, deb, opkg
  - `rpm -ivh my-wonderful-app.rpm` ? (-)
  - Facile (+) mais difficile à maintenir -> gestion des dépendances (-)
  - Non Atomique et non applicable pour l'embarqué (-)
- **Container**
  - Concept intéressant (+)
  - Implique de gérer les applications dans un container (-)

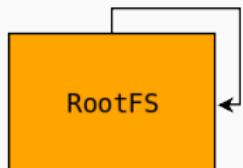
# Sur quelle base ?

- **Fichiers**
  - A éviter, difficile de garantir l'atomicité (-)
- **Gestionnaire de paquets**
  - RPM, deb, opkg
  - `rpm -ivh my-wonderful-app.rpm` ? (-)
  - Facile (+) mais difficile à maintenir -> gestion des dépendances (-)
  - Non Atomique et non applicable pour l'embarqué (-)
- **Container**
  - Concept intéressant (+)
  - Implique de gérer les applications dans un container (-)
- **Delta (atomique)**
  - Faible bande-passante (+)
  - Complexe (-)
  - Risques sur la corruption du système de fichiers principal (-)

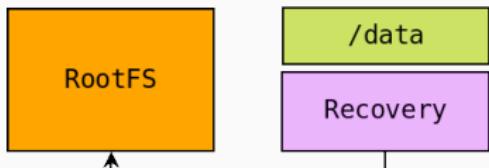
# Sur quelle base ?

- **Fichiers**
  - A éviter, difficile de garantir l'atomicité (-)
- **Gestionnaire de paquets**
  - RPM, deb, opkg
  - rpm -ivh my-wonderful-app.rpm ? (-)
  - Facile (+) mais difficile à maintenir -> gestion des dépendances (-)
  - Non Atomique et non applicable pour l'embarqué (-)
- **Container**
  - Concept intéressant (+)
  - Implique de gérer les applications dans un container (-)
- **Delta (atomique)**
  - Faible bande-passante (+)
  - Complexe (-)
  - Risques sur la corruption du système de fichiers principal (-)
- **Image complète : Full OS**
  - Le cas le plus courant dans l'embarqué
  - Facile à mettre en oeuvre (+)

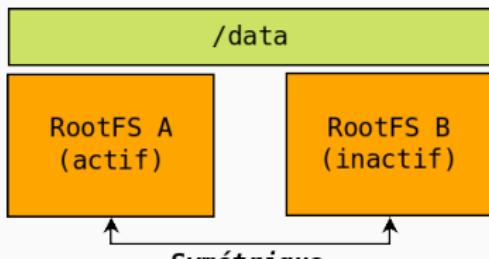
# Quels mécanismes ?



*Run Time*



*Asymétrique*



*Symétrique*

- En fonctionnement :

- Non Atomique (quelques exceptions)
- Downtime très court
- Qui ? : Package managers, AGL

- Mode maintenance : initrd/initramfs

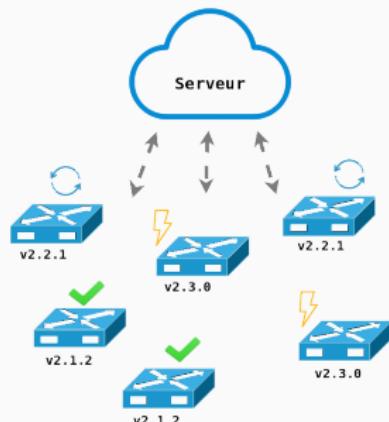
- Robuste
- Retour en arrière impossible si erreur
- Downtime -> long
- Qui ? : Android (avant Nougat)

- A/B ou Seamless update : 

- Robuste, mais coûteux en espace de stockage
- Retour en arrière possible si erreur
- Downtime -> court
- Toujours opérationnel
- Qui ? : Android (depuis Nougat)

# Et comment ?

- Mises à jour sur site
  - Pas de connectivité
  - Accès physique :
    - Gestion par **clé USB/carte SD**
    - Interactif
    - Déplacement d'un technicien €€€
- Mises à jour distantes : OTA
  - Pas d'accès physique :
    - **HTTPS, FTP, ...**
    - Pas d'interaction
  - Serveur pour la gestion des périphériques (flotte)
    - Mise à jour programmée
    - Campagne de mise à jour
    - Inventaire des périphériques
    - Statut des périphériques
    - Version logicielle courante
    - Gestion des artefacts



# Exigences clés d'un dispositif de mise à jour

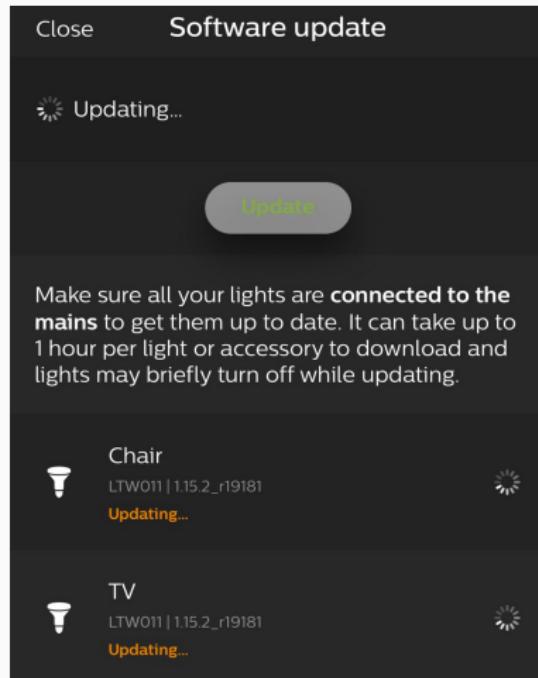


- Doit être capable de mettre à jour l'ensemble des composants
  - *Bootloader = dangereux*
- Doit s'interfacer avec le Bootloader (e.g *Environnement U-Boot*)
- Doit être Robuste (coupures de courant et pertes de connexions réseau)
  - *L'opération doit être atomique = Pas d'installation partielle*
- Ne doit pas rendre le périphérique inutilisable (Fail-safe)
  - Notion de *Rollback*
- Doit disposer d'un espace pour les données persistentes (*Stateless FS*)
  - *Partition dédiée*
- Doit être sécurisé
  - *Empêcher une action non autorisée*
- Doit permettre une gestion Locale et Distante (OTA)
- Doit permettre d'effectuer des tests (*Sanity check*) avant validation

## Downtime

---

# Exemple 1/4



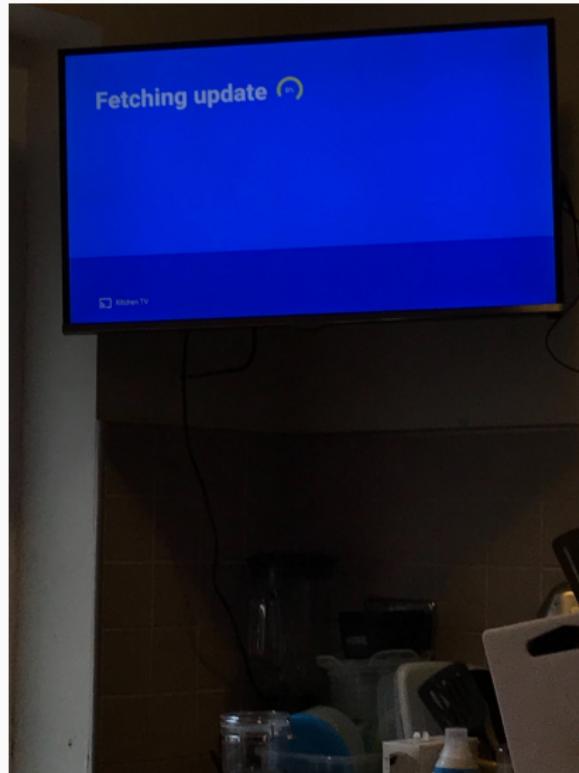
*@internetofshit*

## Exemple 2/4



*@internetofshit*

## Exemple 3/4



@internetofshit

## Exemple 4/4



**Robuste ?!**

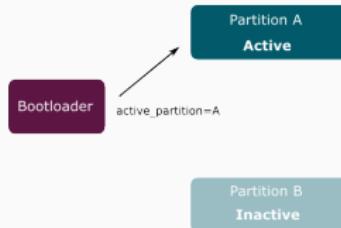
---

# Coupure de courant ?!



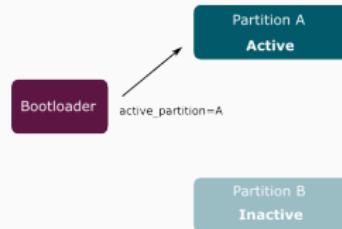
# Atomicité : Exemple schéma A/B

- Démarrage sur A



# Atomicité : Exemple schéma A/B

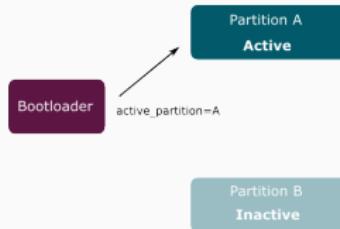
- Démarrage sur A



- Mise à jour de B

# Atomicité : Exemple schéma A/B

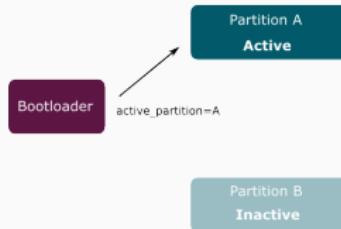
- Démarrage sur A



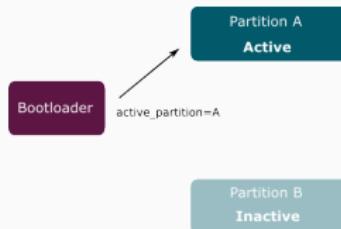
- Mise à jour de B
- coupure de courant !**

# Atomicité : Exemple schéma A/B

- Démarrage sur A

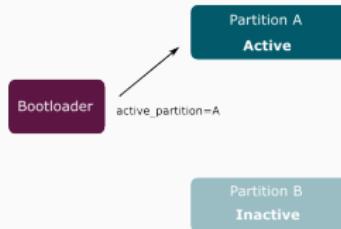


- Mise à jour de B
- coupe de courant !**
- Démarrage sur A

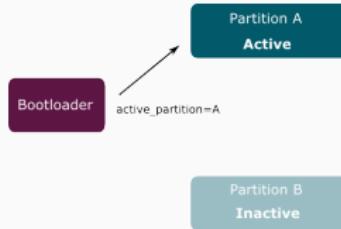


# Atomicité : Exemple schéma A/B

- Démarrage sur A



- Mise à jour de B
- coupe de courant !**
- Démarrage sur A



- Mise à jour de B

# Rollback ?!

---

*"Etre capable de revenir sur une version stable (et fonctionnelle) lors de la détection d'un problème"*

# Pour éviter ...



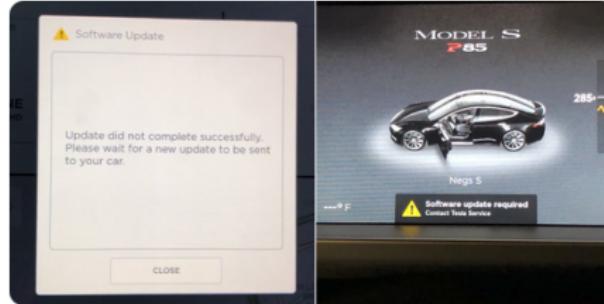
Ryan Negri @RyanNegri

Follow



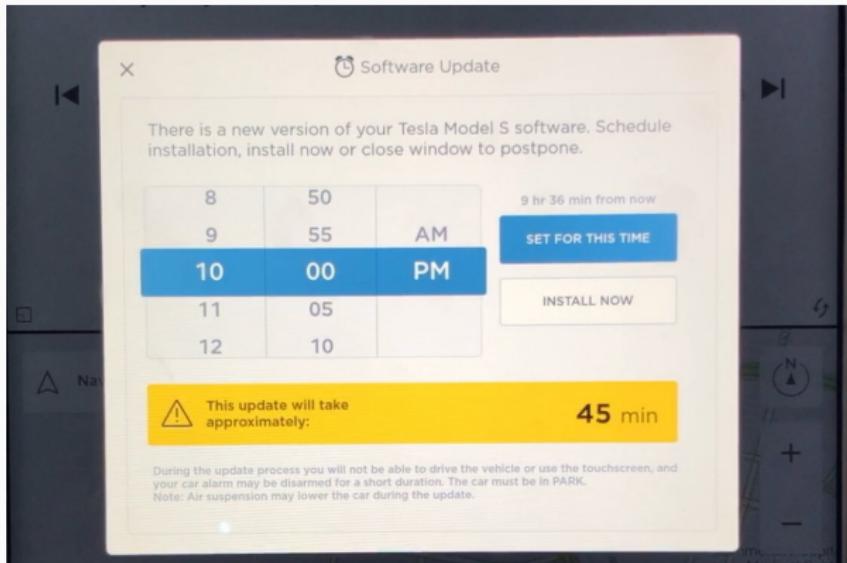
Software update issue with the Model S today. Car is immobile at the moment.

Tesla tech support and roadside assistance have been very helpful - we're hoping to avoid the tow by forcing a new update to the car.



11:47 AM - 5 Sep 2018

# Pour éviter ...



Pour éviter ...

## Update gone wrong leaves 500 smart locks inoperable

Fatal error leaves customers scrambling for fixes that can take a week or longer.

DAN GOODIN - 8/15/2017, 12:07 AM



# Rollback : Quand ?

Plusieurs cas de figures :

- Echec d'une mise à jour
  - DéTECTé par l'updater
- Mise à jour réussie mais problème Noyau (Kernel Panic)
  - BUG logiciel !
- Mise à jour réussie mais problème sur l'applicatif métier
  - BUG logiciel !

## Rollback : Scénario

- Démarrage sur A

## Rollback : Scénario

- Démarrage sur A
- Mise à jour de B
  - Mise à jour du flag `active_partition` (flag bootloader)

## Rollback : Scénario

- Démarrage sur A
- Mise à jour de B
  - Mise à jour du flag `active_partition` (flag bootloader)
- Reboot

## Rollback : Scénario

- Démarrage sur A
- Mise à jour de B
  - Mise à jour du flag `active_partition` (flag bootloader)
- Reboot
- Démarrage sur B
  - **Anomalie** de l'application principale

# Rollback : Scénario

- Démarrage sur A
- Mise à jour de B
  - Mise à jour du flag `active_partition` (flag bootloader)
- Reboot
- Démarrage sur B
  - **Anomalie** de l'application principale
- Détection de l'anomalie
  - Reboot

# Rollback : Scénario

- Démarrage sur A
- Mise à jour de B
  - Mise à jour du flag `active_partition` (flag bootloader)
- Reboot
- Démarrage sur B
  - **Anomalie** de l'application principale
- Détection de l'anomalie
  - Reboot
- Rollback
  - Mise à jour du flag `active_partition` (flag bootloader)

# Rollback : Scénario

- Démarrage sur A
- Mise à jour de B
  - Mise à jour du flag `active_partition` (flag bootloader)
- Reboot
- Démarrage sur B
  - **Anomalie** de l'application principale
- Détection de l'anomalie
  - Reboot
- Rollback
  - Mise à jour du flag `active_partition` (flag bootloader)
- Démarrage sur A

# Rollback : Fondamentaux 1/2

Gestion des Kernel Panic :

# Rollback : Fondamentaux 1/2

## Gestion des Kernel Panic :

- Utilisation de **CONFIG\_PANIC\_TIMEOUT**
  - Pour définir le timeout avant redémarrage sur un Kernel Panic

# Rollback : Fondamentaux 1/2

## Gestion des Kernel Panic :

- Utilisation de **CONFIG\_PANIC\_TIMEOUT**
  - Pour définir le timeout avant redémarrage sur un Kernel Panic

## "Sanity Check" :

# Rollback : Fondamentaux 1/2

## Gestion des Kernel Panic :

- Utilisation de **CONFIG\_PANIC\_TIMEOUT**
  - Pour définir le timeout avant redémarrage sur un Kernel Panic

## "Sanity Check" :

- Après la mise à jour, il faut vérifier si le système est opérationnel avant de valider celle-ci :

# Rollback : Fondamentaux 1/2

## Gestion des Kernel Panic :

- Utilisation de **CONFIG\_PANIC\_TIMEOUT**
  - Pour définir le timeout avant redémarrage sur un Kernel Panic

## "Sanity Check" :

- Après la mise à jour, il faut vérifier si le système est opérationnel avant de valider celle-ci :
  - Vérification de l'applicatif métier (UI, services, serveur, ...),
  - Vérification de la configuration système,
  - ...

# Rollback : Fondamentaux 1/2

## Gestion des Kernel Panic :

- Utilisation de **CONFIG\_PANIC\_TIMEOUT**
  - Pour définir le timeout avant redémarrage sur un Kernel Panic

## "Sanity Check" :

- Après la mise à jour, il faut vérifier si le système est opérationnel avant de valider celle-ci :
  - Vérification de l'applicatif métier (UI, services, serveur, ...),
  - Vérification de la configuration système,
  - ...

## Mécanisme de redémarrage lors d'une défaillance :

# Rollback : Fondamentaux 1/2

## Gestion des Kernel Panic :

- Utilisation de **CONFIG\_PANIC\_TIMEOUT**
  - Pour définir le timeout avant redémarrage sur un Kernel Panic

## "Sanity Check" :

- Après la mise à jour, il faut vérifier si le système est opérationnel avant de valider celle-ci :
  - Vérification de l'applicatif métier (UI, services, serveur, ...),
  - Vérification de la configuration système,
  - ...

## Mécanisme de redémarrage lors d'une défaillance :

- Watchdog
  - Pour détecter les *freezes*
  - Solution matérielle

## Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

Exemple : `bootlimit=1`

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
    - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
      - `bootcount` : variable incrémentée à chaque redémarrage
      - `bootlimit` : pour définir le nombre maximal de redémarrage
      - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
      - `upgrade_available` : pour la gestion de `bootcount`
  - Interaction Bootloader <-> userspace afin de positionner les variables
    - Utilisation de `fw_setenv` et `fw_printenv`
- Exemple : `bootlimit=1`
- Démarrage sur A, mise à jour de B

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

Exemple : `bootlimit=1`

- Démarrage sur A, mise à jour de B
- `fw_setenv bootcount 0`

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

Exemple : `bootlimit=1`

- Démarrage sur A, mise à jour de B
- `fw_setenv bootcount 0`
- `fw_setenv upgrade_available 1`

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

Exemple : `bootlimit=1`

- Démarrage sur A, mise à jour de B
- `fw_setenv bootcount 0`
- `fw_setenv upgrade_available 1`
- Démarrage sur B (`bootcount=1`)

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

Exemple : `bootlimit=1`

- Démarrage sur A, mise à jour de B
- `fw_setenv bootcount 0`
- `fw_setenv upgrade_available 1`
- Démarrage sur B (`bootcount=1`)
- Si OK `fw_setenv upgrade_available 0`, MAJ validée

# Rollback : Fondamentaux 2/2

Gestion des cycles de redémarrage répétitif (e.g U-Boot) :

- Implémentation Bootcount Limit
  - `CONFIG_BOOTCOUNT_LIMIT` et `CONFIG_BOOTCOUNT_ENV`
    - `bootcount` : variable incrémentée à chaque redémarrage
    - `bootlimit` : pour définir le nombre maximal de redémarrage
    - `altbootcmd` : séquence alternative si `bootcount > bootlimit`
    - `upgrade_available` : pour la gestion de `bootcount`
- Interaction Bootloader <-> userspace afin de positionner les variables
  - Utilisation de `fw_setenv` et `fw_printenv`

Exemple : `bootlimit=1`

- Démarrage sur A, mise à jour de B
- `fw_setenv bootcount 0`
- `fw_setenv upgrade_available 1`
- Démarrage sur B (`bootcount=1`)
- Si OK `fw_setenv upgrade_available 0`, MAJ validée
- Si NOK, redémarrage, `bootcount > bootlimit`, exécution de la commande `altbootcmd`, retour sur A

# Rollback : Gestion du Watchdog

- Interface = /dev/watchdog
- Utilisation via systemd
  - Watchdog matériel :
    - RuntimeWatchdogSec
    - ShutdownWatchdogSec
  - Watchdog logiciel (pour les services) : "*keep-alive ping*"

```
[Unit]
Description=FooBar application

[Service]
ExecStart=/usr/bin/foobarapp
WatchdogSec=10s
Restart=on-failure
StartLimitInterval=5min
StartLimitBurst=2
StartLimitAction=reboot
...
```

## Rollback : protection



- Mécanisme vulnérable aux attaques :
  - Retour sur l'ancienne partition afin d'y exploiter une faille de sécurité
- Utilisation TPM (Trusted Platform Module)
  - Utilisation d'un compteur (*counter*)
  - afin d'éviter qu'un attaquant ne remplace un nouveau logiciel sécurisé par un logiciel vulnérable plus ancien

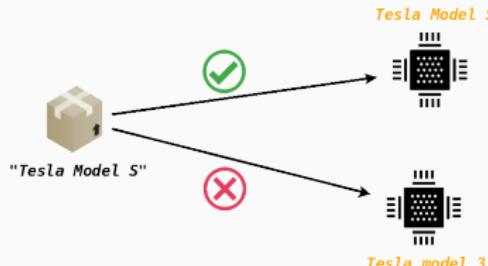
# Sécurité ?!

---

# Comment empêcher une installation non autorisée ?

De façon générale :

- Gestion des compatibilités (Même matériel != Même produit)



- Contrôle d'intégrité

Lié à la sécurité :

- Communication sécurisée (OTA)



- Cryptographie (signature/vérification)



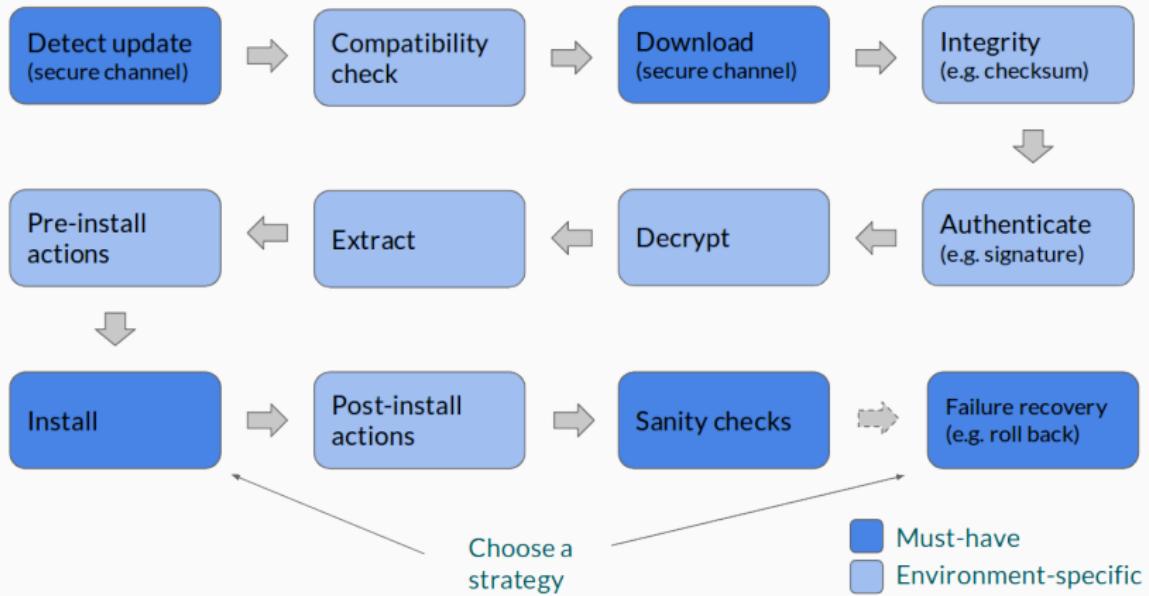
?

My sister-in-law's husband just got this USB  
in the mail for his Jeep. Cc:  
[@0xcharlie](mailto:@nudehaberdasher)



9:36 AM - 3 Sep 2015

# Pour résumer ...



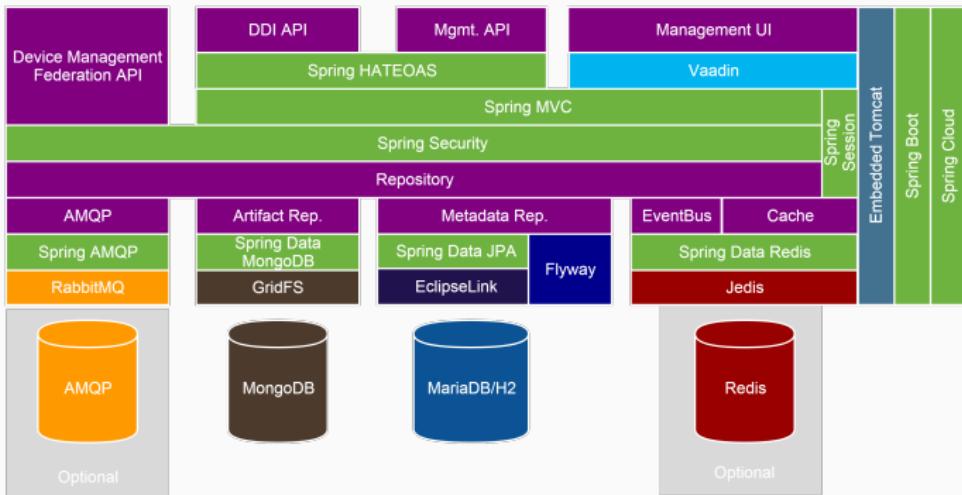
*mender.io*

## **Les différents projets Open-Source**

---



**Eclipse hawkBit is a domain independent back-end framework for rolling out software updates**



# Frameworks

---



## Robust Auto-Update Controller

- <https://github.com/rauc>
- GPLv2.1
- C & GLib
- Client très léger (167.7Ko)
- Mise à jour Symétrique/Asymétrique + flexibilité (data, appfs, ...)
  - Configurable par le fichier `system.conf`
  - Notion de `Slot`
- U-Boot, Barebox, GRUB, EFI
- Support `PKCS#11` (HSM)
- SD/eMMC, UBI, Raw NAND, SquashFS
- HTTP(S), FTP, SSH

# RAUC

- "Bundle" au format SquashFS et signé (**x509**)
- Rollback non intégré (exemple `rauc/contrib/u-boot.sh`)
  - Bootchooser Framework
  - `BOOT_ORDER`, `BOOT_bootname_LEFT`
  - `rauc status mark-good` & `rauc status mark-bad`
  - Utilisation des scripts U-Boot
- Compatible casync (delta update)
- Contrôlable via **D-Bus**
- Notion de **handlers** et **hooks** -> customisation
- Mise à jour locale (USB) ou distante (rauc-hawktail)
- Bien documenté mais ...
  - Très peu d'exemples pour l'intégration
- Build System



**yocto**  
PROJECT





## SWUpdate: software update for embedded system

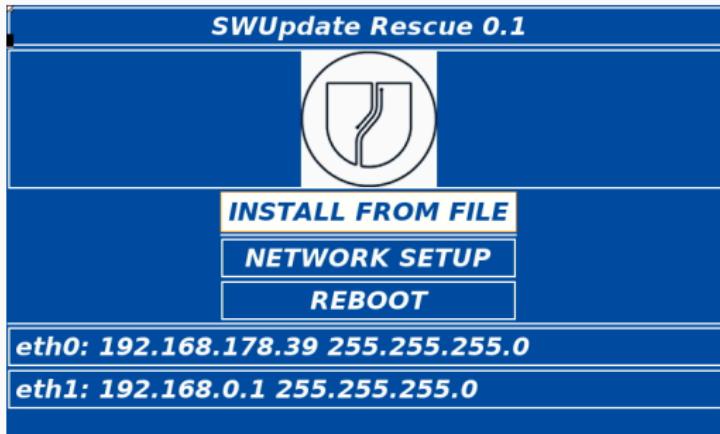
- <https://github.com/sbabic/swupdate>
- Un mix de MIT, GPLv2 and GPLv2+
- C
- Client léger (400Ko)
- Mise à jour Symétrique/Asymétrique + flexibilité
  - Configurable par le fichier **sw-description**
  - XML, JSON et **libconfig** ▶ Exemple
- U-Boot, GRUB, EFI
- SD/eMMC, UBI, Raw NAND, NOR/SPI NOR
- HTTP(S), FTP, SSH -> (**curl**) pour l'aspect **networking**

# SWUpdate

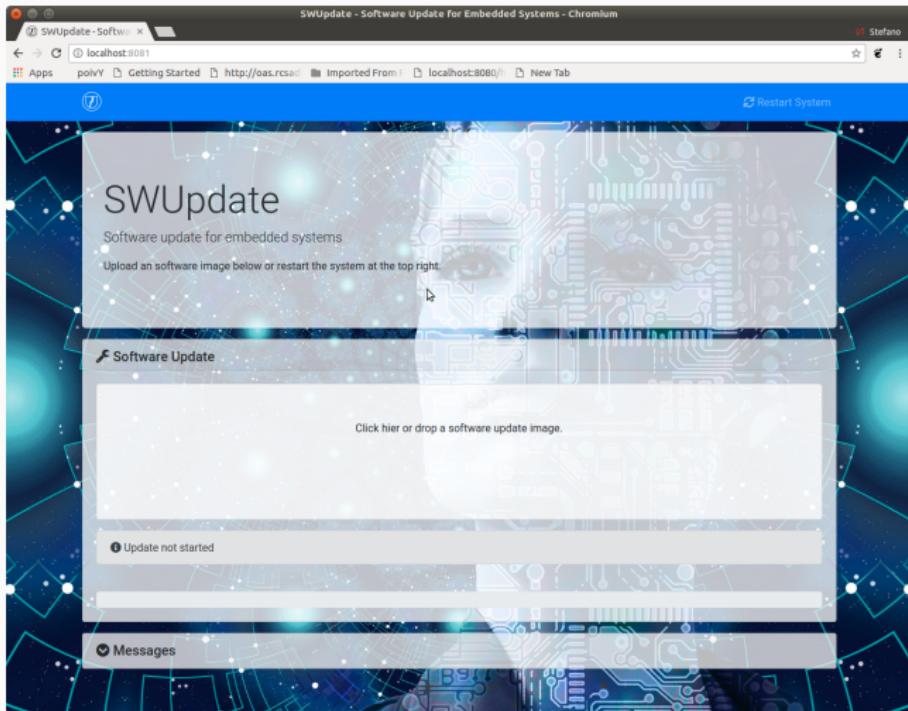
- Archive CPIO (signature et chiffrement possible)
- Rollback non intégré (quelques implémentations disponibles)
- IPC : **Unix Domain Socket** pour avoir le statut (progression)
- Mise à jour locale (USB et serveur web) ou distante (Hawkbit)
- Interface graphique en Lua pour le mode recovery (**RescueGUI**)
- Gestion scripts *pre/post install*
- Gestion microcontrôleur en UART (**ucfw**), gestion maître/esclave (**SWU forwarder**)
- Bien documenté
- Exemples sur `raspberry-pi`, `beaglebone`, `wandboard` -> (`meta-swupdate-boards`)
- Build System



# SWUpdate : RescueGUI



# SWUpdate : Serveur Web



## **Solutions End-to-End**

---



Mender is an end-to-end open source update manager for IoT

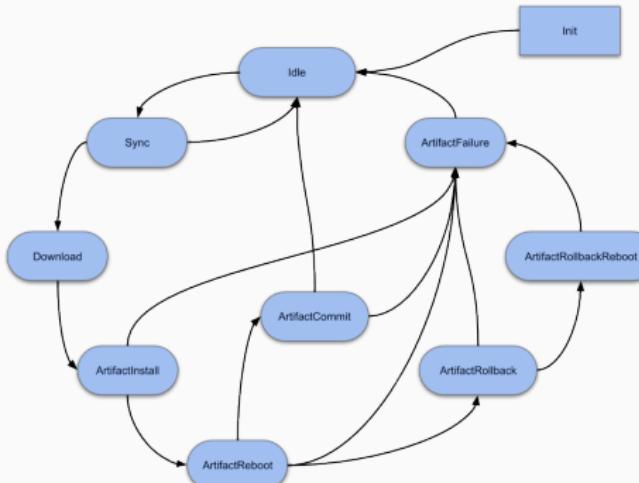
- <https://github.com/mendersoftware>
- Apache 2.0
- Solution **Clé en main** (client + serveur)
- Client lourd (4.6Mo)
- Mise à jour Symétrique seulement
  - 4 Partitions (Boot + RootFS A + RootFS B + Data)
- U-Boot, GRUB
- SD/eMMC, UBI, Raw NAND, NOR
- HTTP(S)

# Mender

- Archive TAR .mender (gestion signature via l'outil mender-artifact)
- Rollback intégré
- State Scripts pour la gestion des actions durant l'opération de mise à jour
- Mise à jour locale (mode standalone) ou distante (Managed)
  - Hosted Mender (payant)
  - Infrastructure Docker
- Facile d'intégration (Auto-patch)
- mender -commit pour valider la mise à jour en mode standalone
- Bien documenté
- Exemples sur <https://hub.mender.io/>
- Build System



# Mender : State Scripts



- Use cases :
  - Migration des données
  - Confirmation utilisateur (*retry later*)
  - ...

# Mender : Hosted Mender

The screenshot shows the Hosted Mender web interface. On the left, there's a sidebar with sections for DASHBOARD, DEVICES, ARTIFACTS, and DEPLOYMENTS. Under DEVICES, a 'Device groups' tab is selected, showing a list of groups: All devices, Borne (selected), and CDL. A button '+ Create a group' is also present. The main content area is titled 'CDL' and displays a table with one row:

Device ID	Device type	Current software	Last heartbeat
5bc33f4fd93e000012b394f	raspi7w-warp	CDL release-1	2018-10-28 16:04

Below the table, there's a section for 'Device public key' with a long base64 string. Further down are sections for 'Device identity' and 'Device inventory', each with a table showing device details like IP address, MAC address, and network interfaces.

▶ hosted mender

# Mender : Mender Hub

 MENDER HUB

## Microchip SAMA5d27-SOM1-EK1 🔒

Board Integrations Yocto Project sumo yocto atmel sama5 microchip

 texierp 3 View history 1d

### Board description

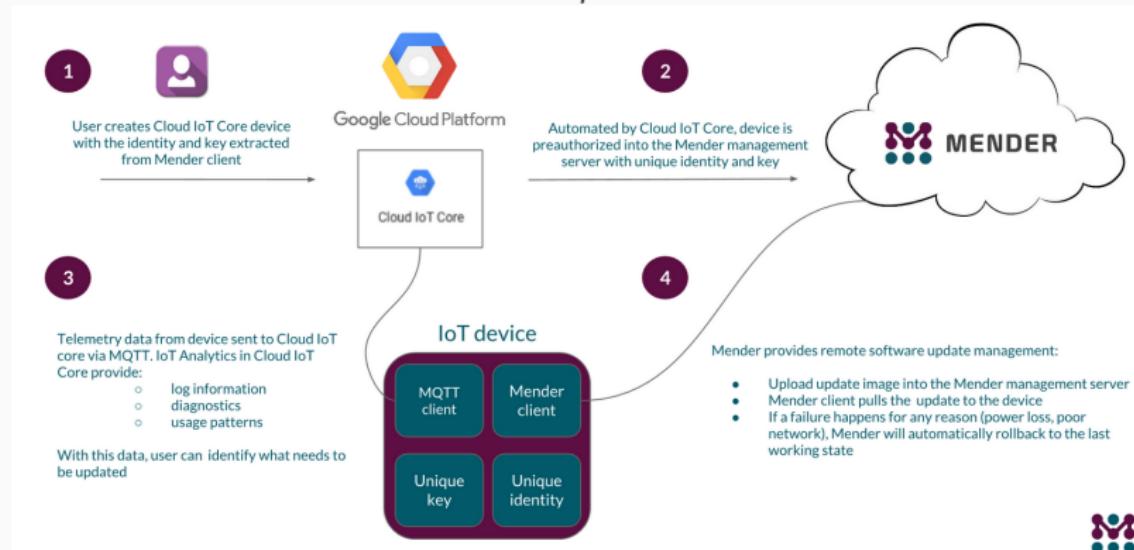
The SAMA5D27-SOM1-EK1 is a fast prototyping and evaluation platform for the SAMA5D2 based System in Packages (SiPs) and the SAMA5D27-SOM1 (SAMA5D27 System On Module). The kit comprises a baseboard with a soldered ATSAM5D27-SOM1 module. The module features an ATSAMA5D27C-D1G-CU SIP embedding a 1-Gbit (128 MB) DDR2 DRAM



URL: <https://www.microchip.com/developmenttools/ProductDetails/atsama5d27-som1-ek1>

# Mender : GCP/Mender

*"Google Cloud IoT Core integrates with Mender.io to analyze issues and enable OTA software updates for IoT"*



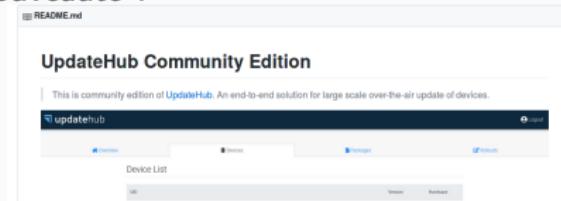


## An end-to-end solution for large scale over-the-air update of devices

- <https://github.com/updatehub>
- GPLv2
- Solution **Clé en main** (client + serveur)
- Golang (mais passage prévu en Rust !)
- Client léger (476ko)
- Mise à jour Symétrique/Asymétrique + flexibilité
  - Configurable par le fichier `cible_uhupkg.config` ▶ Exemple
- U-Boot, GRUB
- SD/eMMC, UBI, Raw NAND, NOR
- HTTP(S)

# UpdateHub

- "Package" signé
- Rollback intégré
- Contrôlable via plusieurs SDK : **Go, Qt, Python**
  - **Unix Domain Socket**
- Mise à jour locale (USB) ou distante (Serveur UpdateHub -> €)
  - Bientôt une nouveauté ?



- Bien documenté
- Exemples sur raspberry-pi, beaglebone, wandboard, WaRP7, ...
- Build System

**yocto** •  
PROJECT

# UpdateHub : Serveur

The screenshot shows the UpdateHub Server interface. On the left, there's a sidebar with a blue circular icon containing 'CO' and the word 'community'. Below it are four menu items: 'OVERVIEW' (selected), 'ROLLOUTS', 'DEVICES', and 'PACKAGES'. At the bottom of the sidebar is a button labeled '▶ UpdateHub'.

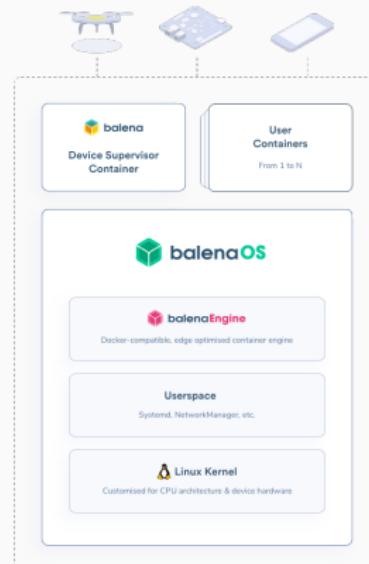
The main area has a dark header with 'Products' and 'Overview' on the left, and a user profile 'Pierre-jean TEXIER' on the right.

The central part of the screen displays a 'Rollouts' section for 'Rollout 2.5-build-b20'. It shows an overall progress bar at 0% with 0% updated, 0% failed, and 100% remaining. Below the bar, two status boxes are shown: 'Pilote' (Running) with 0% Success, 0% Failed, and > 100% Remaining; and 'Production' (Pending) with the same metrics. A large purple circle indicates the status of the rollout.

Below the rollout section are two more sections: 'Devices' (with a donut chart showing 50% for 2.5-build-b22 and 50% for 2.5-build-b07) and 'Packages' (listing two packages: 2.5-build-b22 and 2.5-build-b21, each with their file size and number of artifacts).

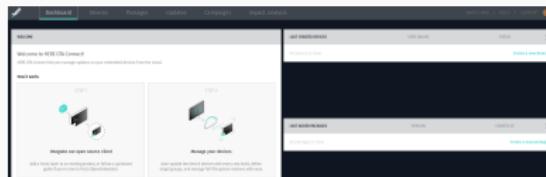
# Mais aussi 1/2

- [resin.io](https://resin.io) maintenant **balena** (Container)
  - Container Utilisateur (docker) (Delta Update -> [librsync](#))
  - Mise à jour symétrique pour la partie **core balenaOS**
    - Read-Only rootFS
  - Integration Yocto/OE = `meta-balena`
  - Supporte seulement : SD/eMMC



# Mais aussi 2/2

- libostree (Delta)
  - *"Like git but for a root file system"*
    - commit = rootfs
  - 1 repository = /ostree/repo et un ensemble de "déploiements" stockés dans /ostree/deploy/...
  - Image initramfs (init.sh)
  - chroot dans /ostree/deploy/myos/234...
  - /usr = lecture seule
  - /var = persistant
  - Utilisé par **AGL**, **Qt OTA**, **Flatpak**, ...
  - Integration Yocto/OE = meta-updater



## RAUC, WaRP7 & Yocto/OE

---

# RAUC : system.conf

- Configuration du système cible : `system.conf`
- Dans `meta-cdl/recipes-core/rauc` (recette dérivée)

```
[system]
compatible=AwesomeProduct
bootloader=u-boot

[keyring]
path=/etc/rauc/ca.cert.pem

[handlers]
post-install=/usr/lib/rauc/post-install.sh

[slot.rootfs.0]
device=/dev/mmcblk1p2
type=ext4
bootname=A

[slot.rootfs.1]
device=/dev/mmcblk1p3
type=ext4
bootname=B
```

# RAUC : Bundle

- Crédit d'un recette permettant la génération du paquet de mise à jour (Bundle) : `cdl-bundle.bb`

```
DESCRIPTION = "RAUC bundle for Capitole du Libre"

inherit bundle

RAUC_BUNDLE_COMPATIBLE = "AwesomeProduct"
RAUC_BUNDLE_VERSION = "v2018-11-17-1"
RAUC_BUNDLE_DESCRIPTION = "Simple Demo Bundle for CDL"
RAUC_BUNDLE_SLOTS = "rootfs"
RAUC_SLOT_rootfs = "core-image-base"
RAUC_SLOT_rootfs[fstype] = "ext4"

RAUC_KEY_FILE = "chemin/vers/development-1.key.pem"
RAUC_CERT_FILE = "chemin/vers/development-1.cert.pem"
```

- Possible aussi avec l'outil `tmp/deploy/tools/rauc`
  - `bitbake rauc-native`

# RAUC : Partitions

- Cr閐ation du fichier **rauc.wks** pour la d閝inition des partitions

```
part /boot --source bootimg-partition --ondisk mmcblk --fstype=vfat \
      --label boot --active --align 4096 --size 20

part / --source rootfs --ondisk mmcblk --fstype=ext4 --label rootfs_A \
      --align 8192 --extra-space 200

part / --source rootfs --ondisk mmcblk --fstype=ext4 --label rootfs_B \
      --align 8192 --extra-space 200

part /data --ondisk mmcblk --fstype=ext4 --label data --size 128M \
       --align 8192 --extra-space 0

bootloader --ptable msdos
```

- Kickstart file pour la cr閐ation des 4 partitions eMMC

# RAUC : Paramètres de l'image

- Utilisation de systemd :

```
# Use systemd as init system
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
VIRTUAL-RUNTIME_initscripts = ""
DISTRO_FEATURES_append = " systemd wifi"
```

- Intégration des éléments vitaux :

```
CORE_IMAGE_EXTRA_INSTALL_append = " rauc rauc-hawkbit \
    packagegroup-base \
    u-boot-fw-utils \
    kernel-devicetree \
"
# Script pour la gestion du rollback
IMAGE_BOOT_FILES_append = " boot.scr"
```

- Encore un peu de configuration :

```
IMAGE_FSTYPES_append = " ext4"
WKS_FILES_imx7s-warp = "rauc.wks"
```

# RAUC : génération

```
$: bitbake cdl-bundle
```

- core-image-base-imx7s-warp.ext4
- core-image-base-imx7s-warp.wic.gz
- cdl-bundle-imx7s-warp.raucb



## Mender, WaRP7 & Yocto/OE

---

# Mender : Configuration 1/2

- Intégration avec U-Boot

```
INHERIT += "mender-full"
MENDER_FEATURES_DISABLE_append = " mender-grub mender-image-uefi"

# Pérophérique de stockage
MENDER_STORAGE_DEVICE = "/dev/mmcblk1"

# Pérophérique de stockage (U-Boot)
MENDER_UBOOT_STORAGE_INTERFACE = "mmc"
MENDER_UBOOT_STORAGE_DEVICE = "0"

# Script U-Boot
IMAGE_BOOT_FILES_append = " boot.scr"
```

# Mender : Configuration 2/2

- meta-cdl/recipes-bsp/u-boot/u-boot-mender-warp7.inc

```
FILESEXTRAPATHS_prepend := "${THISDIR}/u-boot-fslc:"  
  
SRC_URI_remove = " \  
    file://0005-fw_env_main.c-Fix-incorrect-size-for-malloc-ed-strin.patch  
"  
  
SRC_URI_append = " \  
    file://0001-warp7-add-mender-requirements.patch \  
"  
  
MENDER_UBOOT_AUTO_CONFIGURE = "0"  
BOOTENV_SIZE = "0x2000"
```

- Ajout des variables **CONFIG\_BOOTCOUNT\_ENV** et **CONFIG\_BOOTCOUNT\_LIMIT**

```
+CONFIG_BOOTCOUNT_LIMIT=y  
+CONFIG_BOOTCOUNT_ENV=y
```

# Mender : Paramètres de l'image

- Utilisation de systemd :

```
# Use systemd as init system
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
VIRTUAL-RUNTIME_initscripts = ""
DISTRO_FEATURES_append = " systemd wifi"
```

- Configuration Artefact et Serveur :

```
MENDER_ARTIFACT_NAME = "CDL-release-1"
MENDER_SERVER_URL = "https://hosted.mender.io"
MENDER_TENANT_TOKEN = "..."
```

# Mender : génération

```
$: bitbake core-image-base
```

- core-image-base-imx7s-warp.sdimg
- core-image-base-imx7s-warp.mender
- Signature de l'artefact :

```
$: ./mender-artifact \  
    sign core-image-base-imx7s-warp.mender \  
    -k ./private.key
```



# **Buildroot**

---

# Mender

```
make menuconfig
/home/pjtexier/Documents/buildroot/.config - Buildroot 2018.11-rc1-00021-gebffc
a→ Target packages → System tools
mender
BR2_PACKAGE_MENDER:
Mender is an open source over-the-air (OTA) software updater
for embedded Linux devices. Mender comprises a client
running at the embedded device, as well as a server that
manages deployments across many devices.
https://github.com/mendersoftware/mender
Symbol: BR2_PACKAGE_MENDER [=y]
Type : bool
Prompt: mender
Location:
    -> Target packages
    -> System tools
Defined at package/mender/Config.in:1
( 66%)
< Exit >
```

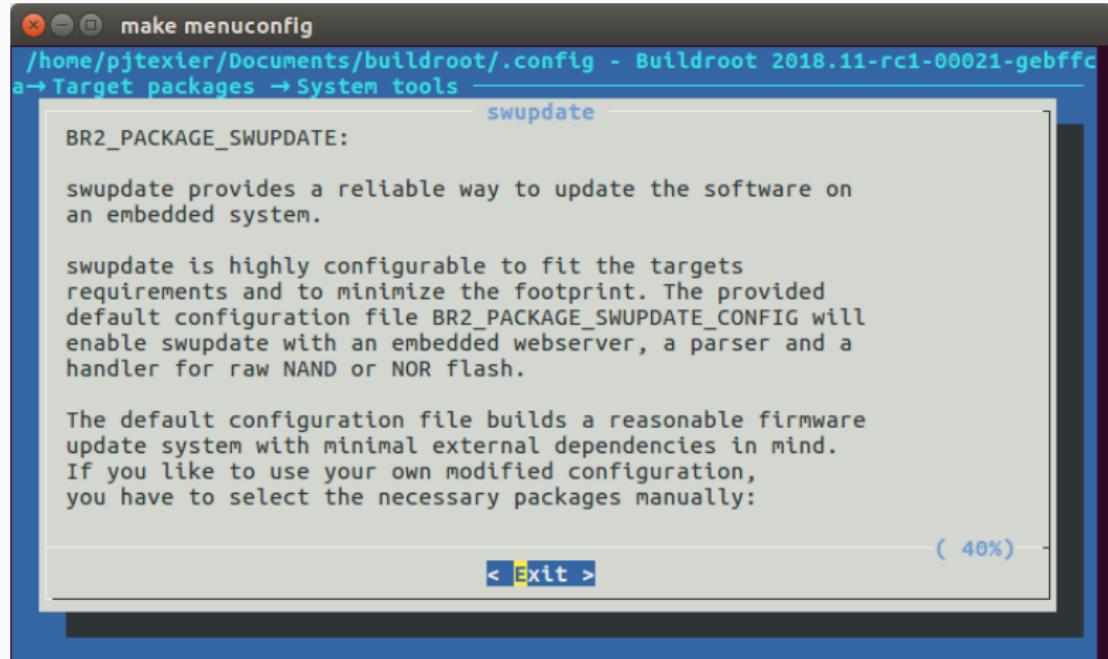
# RAUC

```
make menuconfig
/home/pjtexier/Documents/buildroot/.config - Buildroot 2018.11-rc1-00021-gebffc
a→ Target packages → System tools
                                rauc
BR2_PACKAGE_RAUC:
RAUC is the Robust Auto-Update Controller developed by
Pengutronix. It supports updating embedded Linux
systems over the network or from disks and provides a
d-bus interface.

http://rauc.io/

Symbol: BR2_PACKAGE_RAUC [=y]
Type : bool
Prompt: rauc
Location:
    -> Target packages
    -> System tools
Defined at package/rauc/Config.in:1
( 66%)
< Exit >
```

# SWUpdate



## Démos

---

# Démos

- Mender Mode Standalone
- Mender Mode Hosted Mender
- RAUC

# Conclusion

---

# Conclusion

## Ce qu'il faut retenir :

- Un mécanisme de mise à jour est obligatoire (le penser au début du projet)
- Toujours déployer un logiciel bien testé
- Bien tester les différentes solutions
- Oublier les solutions "Maison"
  - FOSS

**Questions?**