FOTA (Firmware Over the Air)

Frédéric Camps
fcamps@laas.fr

Agenda

Quels sont les enjeux d'un parc IoT distant ?

FOTA / SOTA

Les systèmes cibles

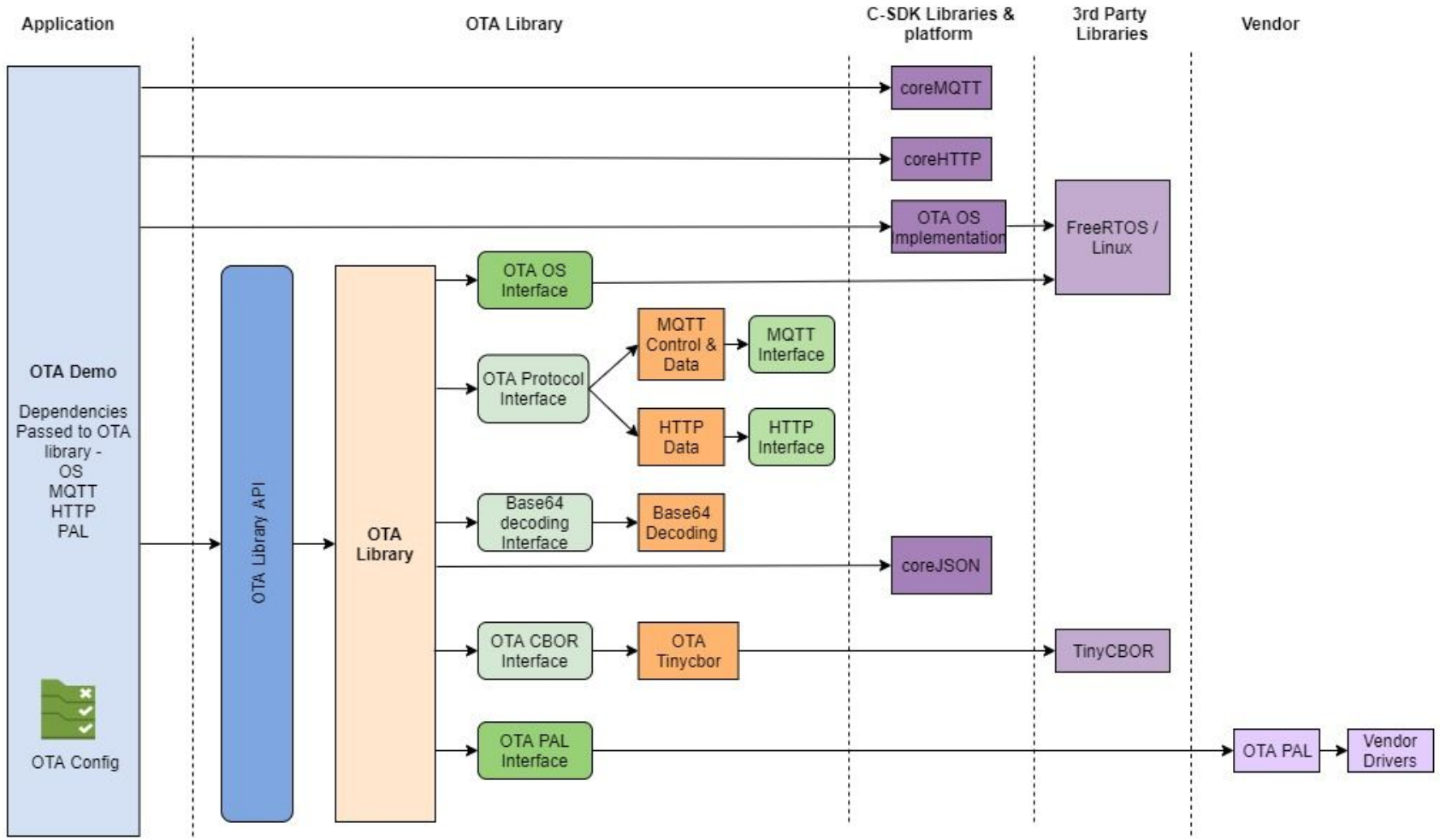
- **Les systèmes avec OS**
- **Les systèmes sans OS**

Les problèmes du FOTA

Architecture des systèmes

https://aws.github.io/ota-for-aws-iot-embedded-sdk/v3.4.0/ota_design.html
https://aws.github.io/ota-for-aws-iot-embedded-sdk/v3.4.0/ota_design.html
https://www.freertos.org/ota/index.html

Exemple : FOTA AWS



Les enjeux du pilotage

Quels sont les enjeux d'un parc IoT distant ?

- Un parc IoT peut être remis à jour:
  - Bugs logiciels et OS,
  - Problème de sécurité,
  - Nouvelles fonctionnalités,
  - Recyclage …
- Impossibilité de déplacement pour les parc trop grands (coût), trop distants, le gestionnaire n'est pas l'exploitant.
- Éviter des disparités dans les versions des systèmes …

Objectifs FOTA

FOTA (Firmware Over the Air) / SOTA (Software Over the Air)

- FOTA pour une mise à jour complète, SOTA le logiciel uniquement,
- Très utilisé en téléphonie mobile, automotive, les box Internet, TV connecté …
- Mise à jour différentes en fonction du besoin:
  - OS: mise à jour la plus lourde,
  - Firmware complet,
  - Fichier de configuration,
  - Librairie,
  - Driver.

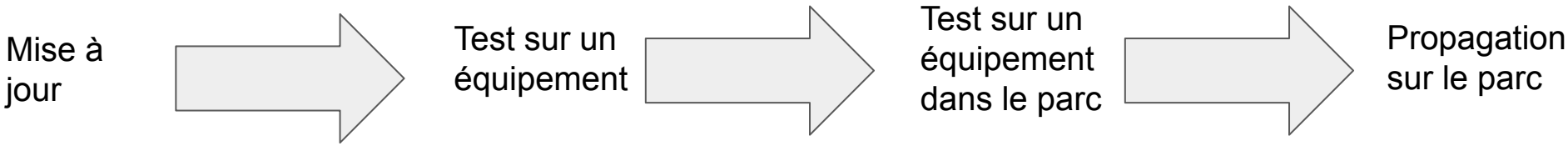
Objectifs FOTA

FOTA (Firmware Over the Air)

- **Cryptage et sécurité :** Les mises à jour FOTA doivent être sécurisées pour éviter toute modification non autorisée du firmware.
- **Redondance et validation :** Les mises à jour FOTA peuvent échouer pour diverses raisons, notamment une perte de connexion ou une interruption de l'alimentation. Les systèmes FOTA intègrent souvent des mécanismes de redondance et de validation pour garantir que la mise à jour est complète et correcte.
- **Planification des mises à jour :** Les mises à jour FOTA peuvent être programmées pour se produire à des moments où l'appareil est moins susceptible d'être utilisé intensivement, minimisant ainsi les interruptions pour les utilisateurs.
- **Gestion de la bande passante**
- **Personnalisation des mises à jour :** Dans certains cas, les fabricants peuvent personnaliser les mises à jour en fonction des besoins spécifiques de chaque appareil ou groupe d'appareils.

Objectifs FOTA

FOTA (Firmware Over the Air)



Architecture des systèmes

Les systèmes cibles

- Les systèmes avec OS (Linux, FreeRTOS …),
- Les systèmes sans OS (microcontrôleur),

Support de la mémoire permanente

- Les systèmes en mémoire permanente (NAND, NOR),
- Les systèmes sur support disque (HDD, SSD, SD card).

Les systèmes avec OS (Linux, Android, Win …)

- Mise à jour via un processus interne à l'OS déjà prévu (FreeRTOS AWS),
- Possibilité de donner un serveur privé pour la mise à jour,
- La mise à jour est parfois très complexe:
  - Placer le nouveau binaire à exécuter sur un support permanent,
  - Mise à jour majeure: écrasement d'une partition sur un support
  - Changement de lib avec un impact dans les binaires
  - Processus de validation parfois complexe (repasser tous les tests !)
- Autre stratégie: une solution figée !

https://www.balena.io
https://qbee.io/iot-ota/
https://aws.amazon.com/fr/iot-device-management/
https://aws.amazon.com/fr/freertos/
https://docs.aws.amazon.com/freertos/latest/userguide/dev-guide-freertos-libraries.html
https://docs.aws.amazon.com/freertos/latest/userguide/ota-update-library.html

Offre de service FOTA / SOTA

- Une offre de service est disponible(AWS, BalenaOS …),
  - AWS ⇒ FreeRTOS ⇒ TLS v1.2 et PKCS #11

- Possibilité de mettre à jour les systèmes distants avec un système propriétaire,

Architecture des systèmes

https://aws.github.io/ota-for-aws-iot-embedded-sdk/v3.4.0/ota_design.html
https://aws.github.io/ota-for-aws-iot-embedded-sdk/v3.4.0/ota_design.html
https://www.freertos.org/ota/index.html

Exemple : FOTA AWS

- Utilisation de PSoC™ 6 MCU: MCUboot-based basic bootloader
- Flash de la mémoire de la carte et reboot sur le programme
- SDK AWS pour les développements
- HTTP, MQTT pour le chargement du firmware

Les systèmes sans OS (microcontrôleur)

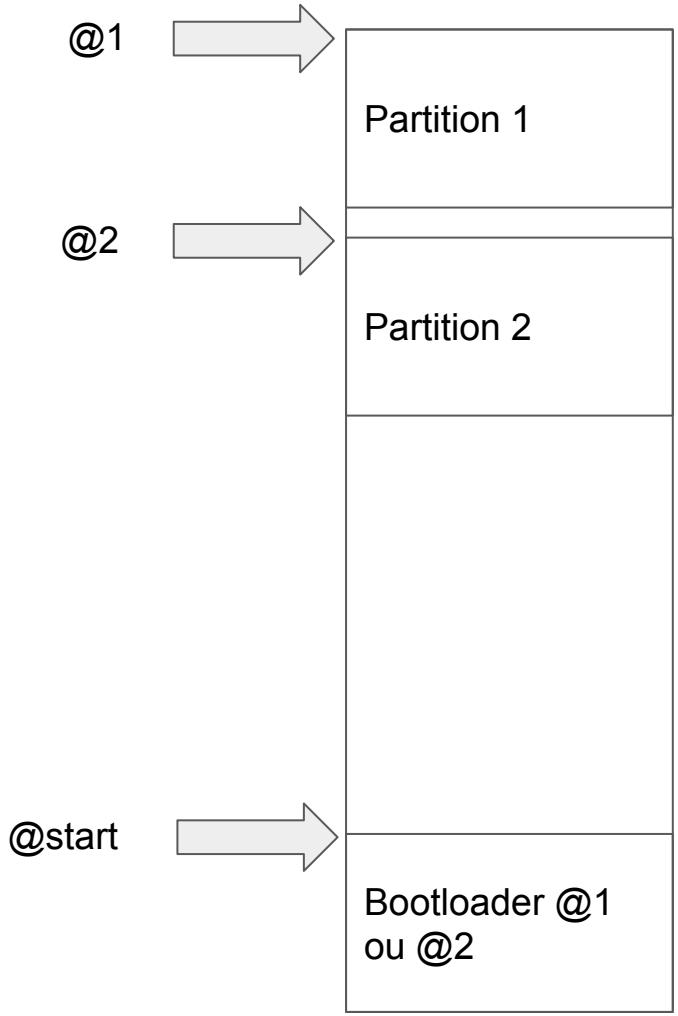
- Mise à jour via un processus de changement de firmware,
- Obligation de tout écraser (ou presque),
- Écrasement d'une partition, ne pas arrêter le processus sous peine de destruction (au mieux retour sur firmware d'usine),
- La mise à jour est parfois très complexe:
  - Chargement, vérification, écrasement d'une partition, reboot
  - Demande un bootloader spécifique qui dépend du microcontrôleur
  - Le bootloader peut gérer plusieurs versions de l'OS

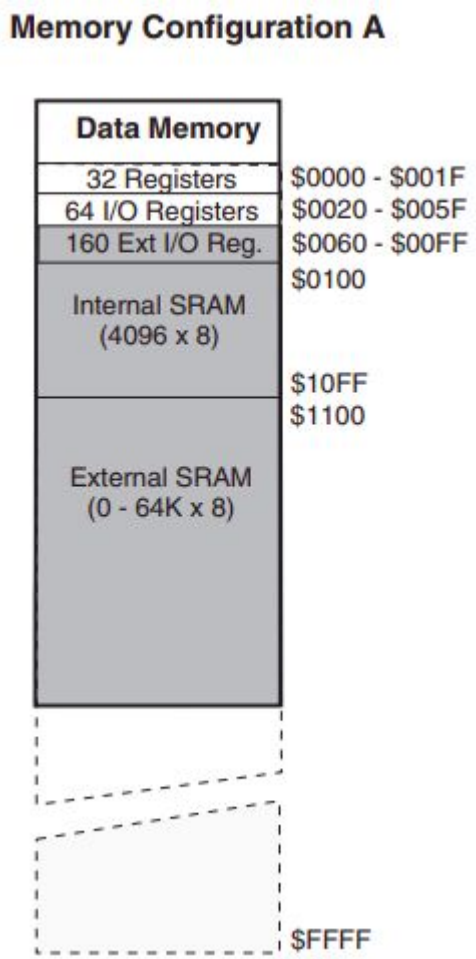
Les systèmes sans OS (microcontrôleur)

- Le microcontrôleur doit avoir une partition suffisante (mémoire flash),
- Il faut développer un bootloader (complexe),
- Il faut un système de chargement du binaire par radio (peut être différent de celui du transport des données habituelles)
- Le redémarrage est conditionnel (succès ou pas, vérification CRC, partition sans problème d'écriture, …)

Architecture des systèmes

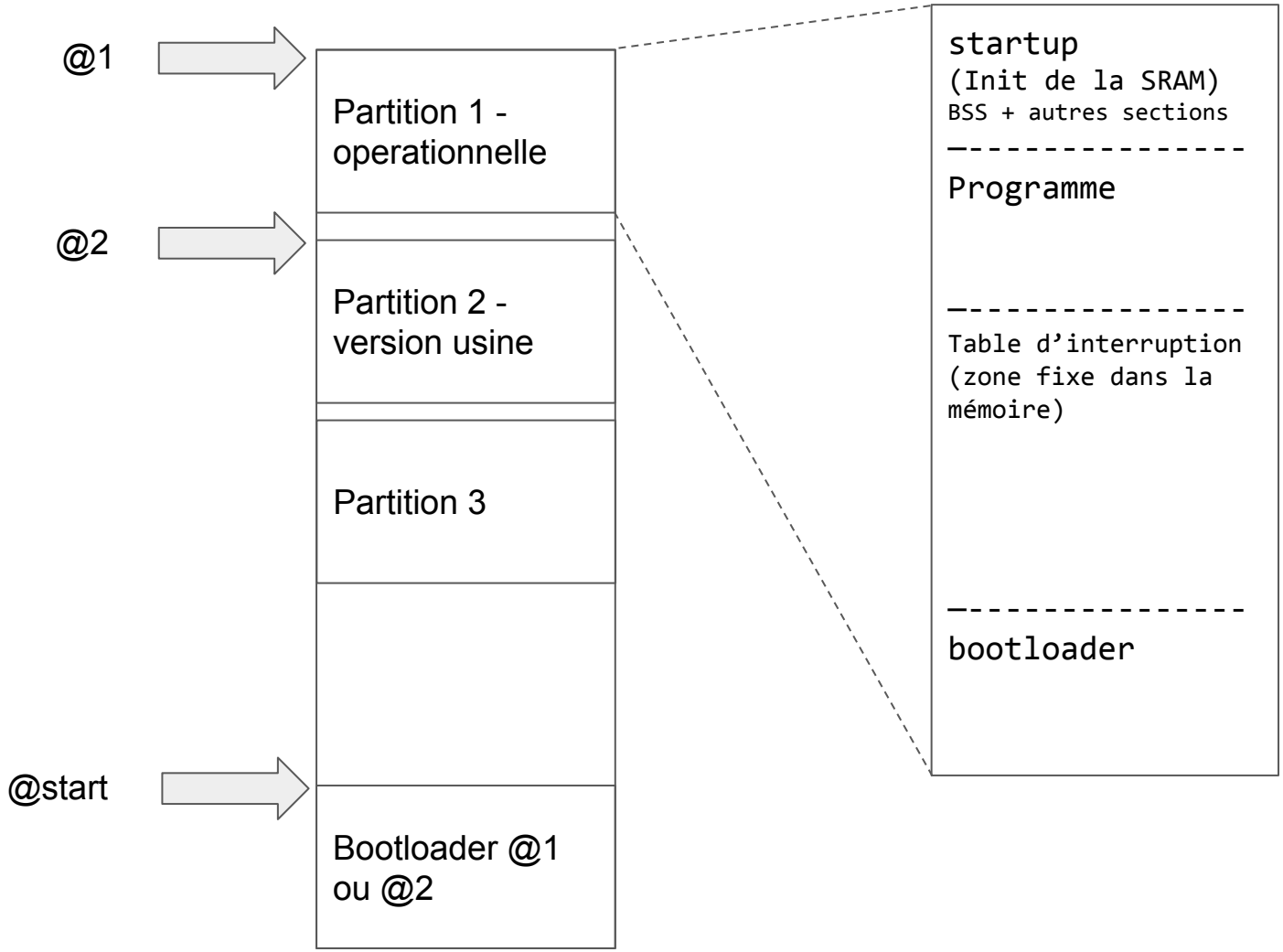
Example ATMEGA128A





Architecture des systèmes

Example ATMEGA128A



Architecture possible pour le FOTA

- Système avec flash interne avec un partitionnement pour les versions du soft → il faut un bootloader
- Flash externe au processeur→ il faut un bootloader
- Double bank de mémoire interne processeur → il faut un bootloader
- Flash sur un autre système -> Il faut flasher avec un jtag ou bootloader

Les problèmes du FOTA

- Utilisation intensive de la batterie,
- Interface 2G, 3G ou 4G LTE pour une bande passante réaliste,
- Un processus complexe (d'une façon générale) mais aussi obligatoire pour les grands parcs,

Les problèmes du FOTA

- Attention certains processeurs ne supportent pas de bootloader,
- Stratégie de mise à jour en même temps que le trafic de service:
  - Dans un paquet on fait passer des morceaux du firmware,
  - Réduction de la consommation de la batterie,
  - Implique une mise à jour au fil de l'eau (assez long).

Architecture des systèmes

Exemple avec un STM32 + ESP8266