open wide
INGÉNIERIE
a Smile group company

Systèmes d'exploitation pour IoT

Pierre Ficheux (pierre.ficheux@openwide.fr)

Avril 2016

open wide INGÉNIERIE
a Smile group company

L'IoT, un réseau d'OS

- L'IOT associe « little data » et « big data »
- Les OS interviennent à plusieurs niveaux
  - Serveurs → UNIX/Linux, Windows ?
  - Terminaux (IHM déportée) → Android, iOS, Linux, Windows
  - Objets (souvent)
- La nécessité de l'OS dépend de la complexité de l'objet
- La fonction de communication (réseau) nécessite le plus souvent un OS et non du « bare metal »
- Le lien entre les OS est la *standardisation* des protocoles et des formats de données

open wide INGÉNIERIE
a Smile group company

Contraintes des OS pour « objets »

- Principaux critères
  - Empreinte mémoire
  - Consommation
  - Stabilité
  - Coût ! (bientôt des milliards d'objets…)
  - Prise en compte de protocoles dédiées (6LoWPAN, ...)
  - Prise en compte du temps réel (?)
- Mais également
  - Évolutivité
  - Portabilité
  - API standards (POSIX, Web) → maintenabilité

OS libre et IoT

- Un OS libre (adapté) respecte les contraintes IoT
- Le choix du logiciel libre favorise l'adoption d'OS comme « service » → focalisation sur la *valeur ajoutée* de l'objet
- Le logiciel libre évite le coût récurrent de la licence
- Développement d'OS dédiés proches des RTOS → *exécutif* temps réel = 1 noyau + 1 application multi-thread
- Certains objets complexes peuvent utiliser des OS comme Android ou GNU/Linux (si optimisés)

open wide INGÉNIERIE
a Smile group company

Quelques exemples d'OS

- RTOS adaptés
  - FreeRTOS
  - Lepton (couche POSIX sur un noyau)
  - eCOS
- OS dédiés
  - TinyOS
  - Contiki
  - RIOT
  - Zephyr (fondation Linux)
- GPOS adaptés
  - GNU/Linux
  - Android & Co

Contiki

- Système d'exploitation développé par le Swedish Institute of Computer Science (SICS, 2002)
  - Licence BSD
  - Ultra léger
  - Flexible
  - Plate-forme d'émulation et de simulation (Cooja)
- Couche réseau uIP et uIPv6
- Optimisé pour la consommation
- Chargement dynamique de modules
- Bien adapté aux capteurs (quelque dizaines de Ko) → à partir des 8 bits (démo OWI sur 8051 datant de... 1980)
- Bonne documentation et nombreux exemples

RIOT

- Démarré en 2008 et maintenu par l'INRIA
- Actuellement en version 2015.12
- Licence GNU LGPL (et non GPL)
- Peut fonctionner avec 1,5 Ko de RAM !
- Temps réel
- Multi-threading complet
- Support C/C++ « standard » très proche de la programmation POSIX classique
- Réseau 6LoWPAN
- CPU 8, 16 et 32 bits
- Présenté par ses concepteurs comme le « Linux de l'IoT »
- Voir l'article dans OS#18

Lepton

- Ajout d'un couche POSIX à un noyau temps réel libre ou propriétaire → portabilité !
- Licence MPL (entre BSD et GPL...)
- Cibles visées 16 ou 32 bits→ Micro-contrôleurs Cortex M4/M3/M0/M0+, ARM9, MIPS…
- 3 axes de développements:
  - Ressources matérielles limitées/basse, consommation
  - Réutilisation
  - Environnement de développement / mise au point
- Disponible sur eCos, Segger embOS, FreeRTOS, ...
- Développé par oloée (Odyssée)

GNU/Linux

- Réservé aux objets complexes
- Fonctionnement sans MMU possible avec µCLinux
- 32 bits + empreinte mémoire de plusieurs Mo
- Licence GPL (pour le noyau)
- Très bon support matériel (OS de référence)
- Pas d'optimisation de la consommation d'énergie
- Distributions classiques (Debian, Ubuntu, …) mal adaptées à l'IoT
- Utilisable grâce à des « build systems » → Buildroot, OpenWrt, Yocto (voir la suite)

Android

- Basé sur un noyau Linux (initialement) modifié par Google
- Équipe plus d'un milliard de téléphones
- Existe déjà sur d'autres objets utilisant une IHM (montres, set-top box)
- Partiellement open source (AOSP), développement non communautaire + quelques pilotes propriétaires
- Empreinte mémoire importante, largement supérieure à celle de GNU/Linux (plusieurs centaines de Mo)
- Pas de build-system, produit une « ROM »
- Développement en Java (simple) + couches basses en C/C++
- Versions dérivées pour les objets (un peu) plus légers (Wear, Brillo)

Distribution Linux

- Ubuntu, Debian, Fedora, etc.
- Environnement bien connu des utilisateurs, simple à aborder, idéal pour débuter mais :
  - Empreinte mémoire importante
  - Long à démarrer + consommation importante
  - Reste un environnement de développement avant tout
  - Faible traçabilité
  - Multi-plateforme souvent limité
- Solution alternative → le « build system »

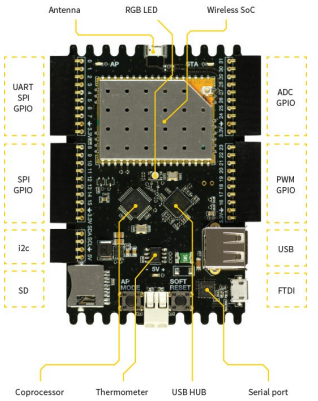
open wide INGÉNIERIE
a Smile group company

Outils de « build system »

- Outil de *création* de distribution
- Crée la distribution à partir des sources des composants en utilisant un « moteur »
- L'outil ne fournit pas les sources mais les *règles de production* et prend en compte les dépendances
- L'outil produit les différents éléments de la distribution
  - Bootloader (si nécessaire)
  - Noyau Linux
  - Images du root-filesystem + applications
- Meilleure solution au niveau empreinte mémoire, consommation, traçabilité, sécurité

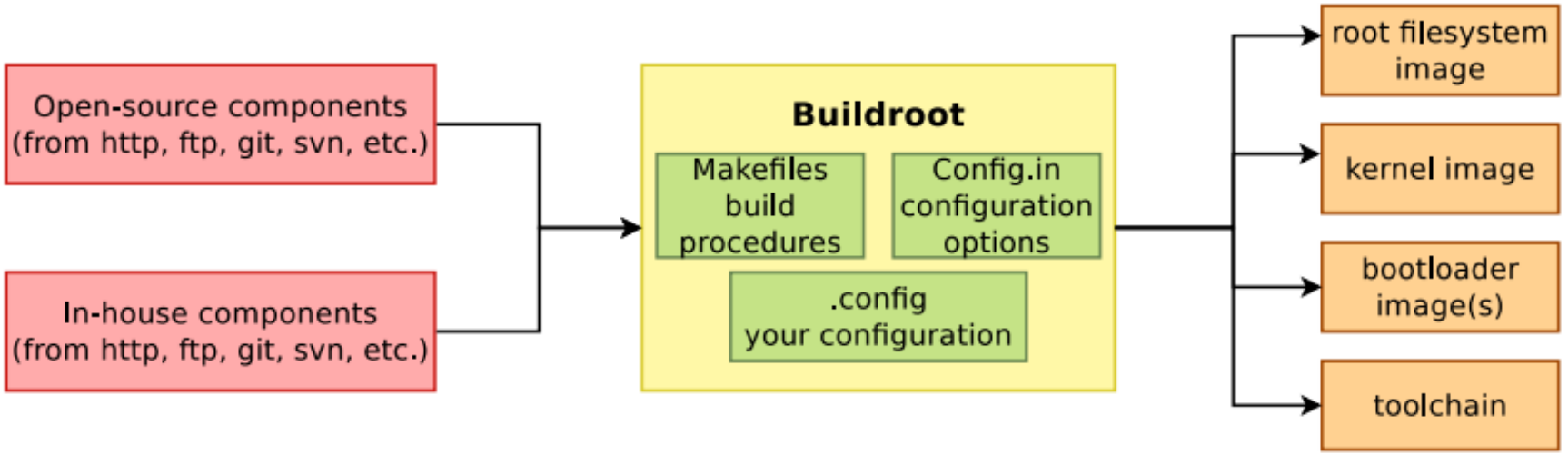
Les principaux outils disponibles

- Yocto/OpenEmbedded
  - Moteur écrit en Python
  - Très puissant mais lourd
  - Basé sur des fichiers de configuration
- Buildroot
  - Basé sur la commande « make »
  - Au départ un démonstrateur pour uClibc
  - Désormais un véritable outil, bien maintenu !
- OpenWrt
  - Dérivé de BR
  - Gère les paquets binaires
  - Utilisé sur WeIO



Buildroot

- Initialement un démonstrateur de uClibc (Micro-C-libC)
- Une version officielle tous les 3 mois depuis 2009.02
- Outil de configuration graphique identique à celui du noyau
- Léger, rapide, basé sur des fichiers `Makefile`
- Pas de système de paquets → « firmware Linux »



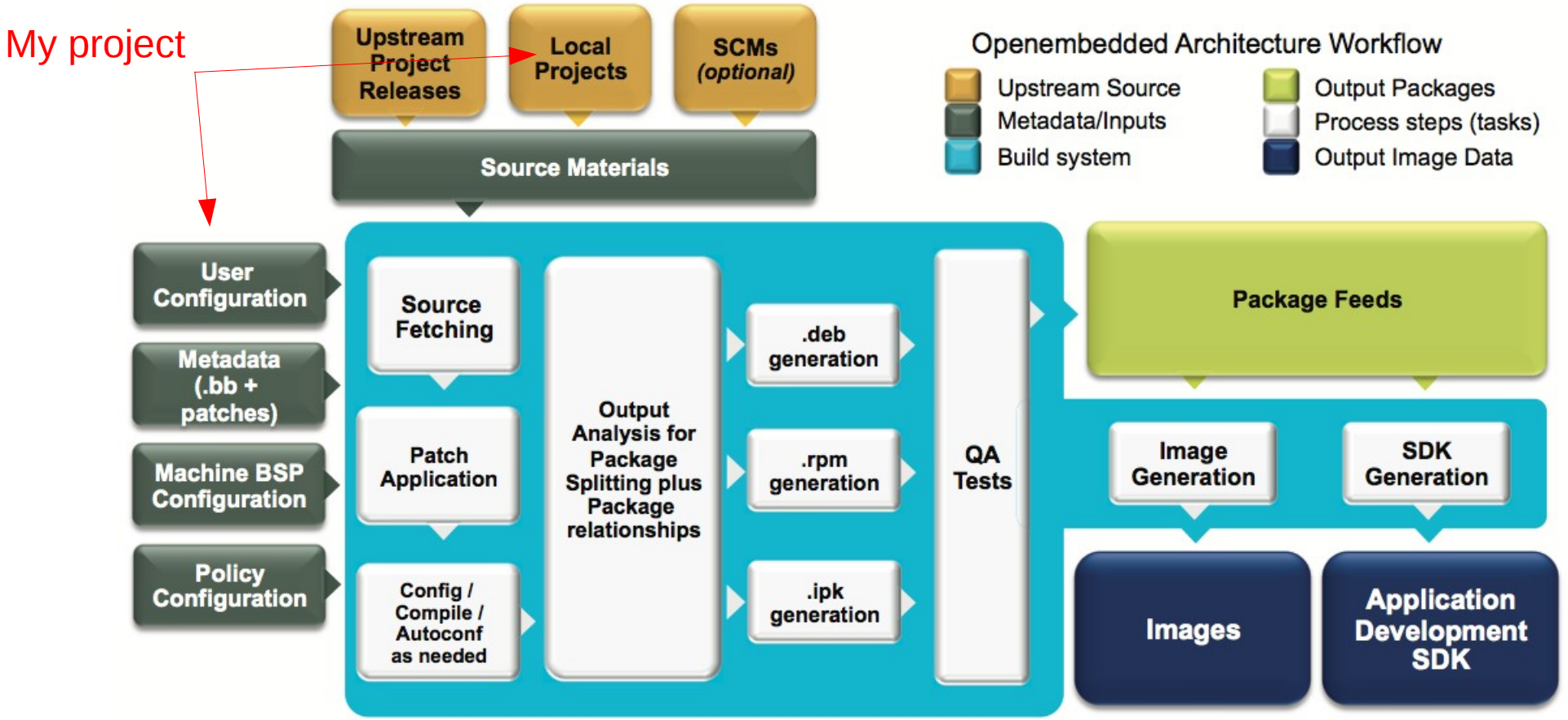
Yocto / OE

- Yocto est un projet « chapeau » démarré en 2010
- Intègre de nombreux projets comme OE, BitBake, *Poky*, Eglibc, ADT, Hob, …
- Plusieurs dizaines de membres dont Intel, Montavista, Freescale, Huawei, Mentor Graphics, Gumstix, ...
- L'architecte est *Richard Purdie* qui a rejoint la *Linux Foundation* en tant que « fellow » en décembre 2010
- Organisation similaire à celle de l'équipe du noyau Linux

  « meritocracy presided over by a benevolent dictator »
- Un *véritable* projet collaboratif promu par la fondation Linux
- De nombreux BSP industriels migrent vers Yocto !

Yocto / OE, suite

- Utilise des paquets RPM, DEB ou IPK
- Syntaxe complexe mais très bonne documentation
- Investissement initial important



Construire un projet

- Construction d'une image de projet sous Buildroot

```
$ make my_project_defconfig
$ make
```

- Idem sous Yocto

```
$ source ./oe-init-build-env my_project
$ bitbake my-project-image
```

- Dans les 2 cas on peut gérer proprement la configuration, donc assurer la traçabilité du projet
- On utilise Yocto si l'on veut une distribution complète et non un « firmware » sans gestion de paquets

open wide INGÉNIERIE
a Smile group company

Buildroot ou Yocto ?

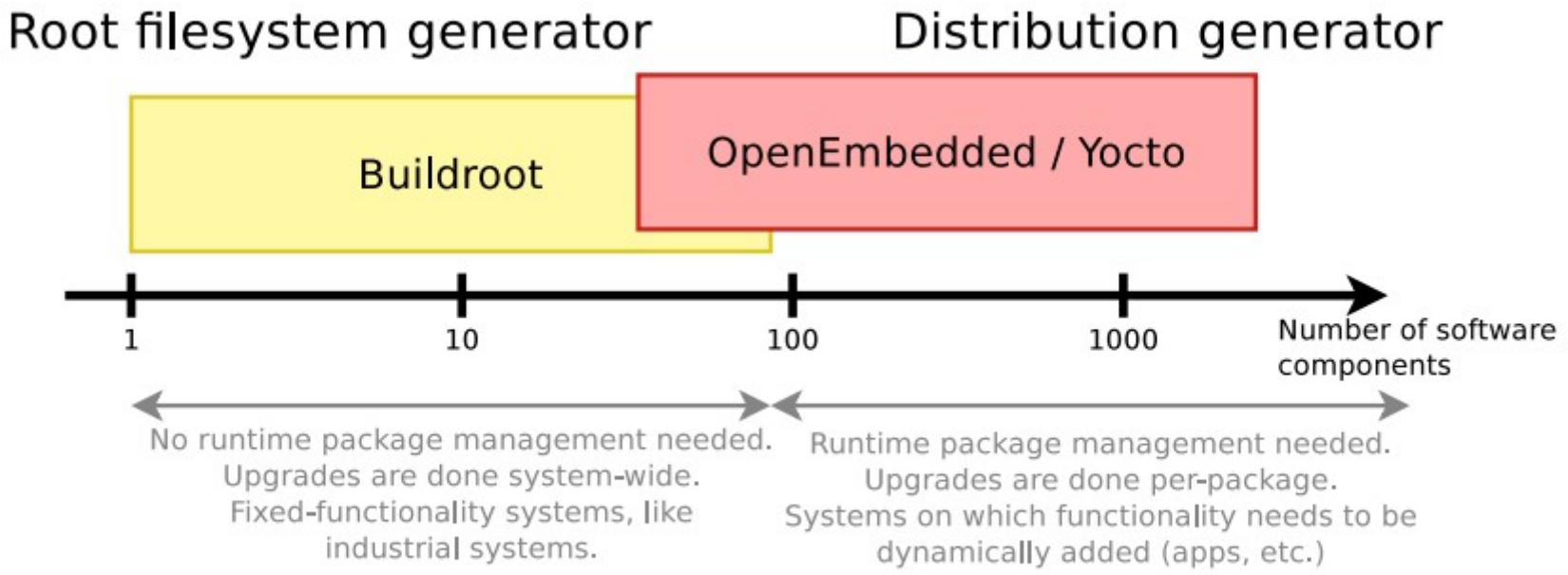


schéma de Thomas Petazzoni

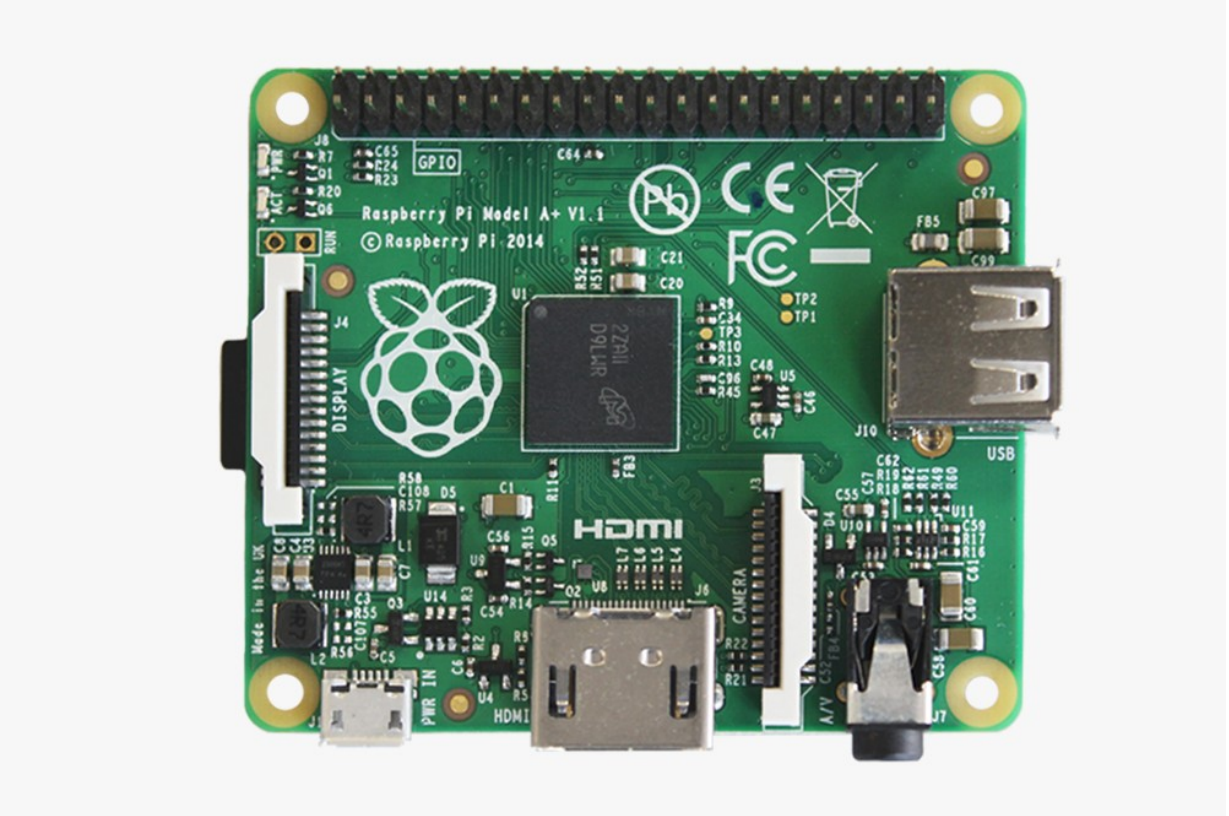
open wide INGÉNIERIE
a Smile group company

Conclusions

- L'utilisation d'un OS libre est fortement conseillée !
- Le choix en OS dédié et OS adapté n'est pas toujours simple
- Les OS dédiés utilisent parfois des API et des outils non standards
- Le panel d'objets est important (de l'automobile au capteur...) il n'y a pas UNE solution universelle
- L'important est la compatibilité des protocoles et la liberté des composants et services essentiels

Démonstration

- Raspberry Pi A+
- Contrôleur Wi-Fi
- Fonctionnement autonome (réseau, relais)
- Configuration, compilation, installation



open wide INGÉNIERIE
a Smile group company

Références

- http://contiki-os.blogspot.fr/2014/02/thingsquares-contiki-iot-workshop.html
- http://www.inria.fr/centre/saclay/actualites/riot-un-os-open-source-pour-l-internet-des-objets
- http://www.riot-os.org
- http://www.riot-os.org/docs/riot-infocom2013-abstract.pdf
- http://ecos.sourceware.org/
- https://source.android.com
- http://elinux.org/Build_Systems
- https://www.yoctoproject.org/
- http://buildroot.uclibc.org
- https://openwrt.org
- http://www.contiki-os.org/start.htmlhttp://we-io.net/hardware
- https://www.zephyrproject.org
- Article « prototypage IoT avec Buildroot » (P. Ficheux, Open Silicium #18)
- Article « RIOT-OS » (Équipe RIOT, Open Silicium #18)
- http://connect.ed-diamond.com/Open-Silicium/OS-007/Lepton-un-systeme-d-exploitation-temps-reel-pour-les-systemes-enfouis2
- http://connect.ed-diamond.com/Open-Silicium/OS-015/Utilisation-de-Lepton-sur-at91samd20