Linux et les logiciels libres pour l'embarqué

Matthieu Herrb

Linux et les logiciels libres pour l'embarqué

Matthieu Herrb

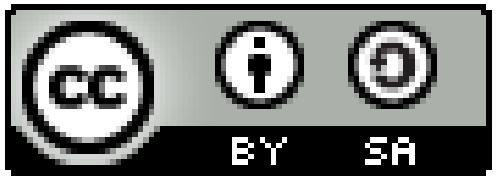
LAAS-CNRS

Université de Toulouse

Envol 2014

http://homepages.laas.fr/matthieu/talks/envol14-l3e.pdf

Licence



Ce document ré-utilise des éléments de :

- la présentation de Pierre Ficheux « état de l'art des systèmes embarqués » aux JDEV 2013.
- les supports de cours de Free Electron « Embedded Linux » et « Yocto Project and OpenEmbedded development »

À propos de l'auteur

Matthieu Herrb
Ingénieur de Recherche au LAAS du CNRS à Toulouse
Responsable de la plateforme robotique
Chargé de la sécurité des systèmes d'information

Activités dans le logiciel libre en dehors de la robotique :

- OpenBSD
- X.Org
- Tetaneutral.net

matthieu.herrb@laas.fr

Agenda

Introduction aux systèmes embarqués

Le logiciel libre dans l'embarqué

Outils de développement pour l'embarqué

Agenda

Introduction aux systèmes embarqués

Le logiciel libre dans l'embarqué

Outils de développement pour l'embarqué

Système / logiciel embarqué

- Association matériel + logiciel
- Tout ce qui n'est pas perçu comme un ordinateur
  - équipements industriels ou scientifiques
  - biens de consommation
- Ciblé : limité aux fonctions pour lesquelles il a été créé
- Fiable et sécurisé : autonomie, processus sensible
- Longue durée de vie
- Optimisé (processeurs moins puissants, contraintes temps-réel)
- Pas toujours un OS

Temps Réel

Le temps intervient dans la correction (validité) du programme :

- réagir en un temps adapté aux événements externes,
- fonctionnement en continu sans réduire le débit du flot d'informations traité,
- temps de calculs connus et modélisables.
- **Latence** maitrisée.

Outils :

- horloges matérielles, interruptions, etc.
- style de programmation adaptés : multi-tâches, évenements,
- approche synchrone (Lustre, Esterel,...)
- langages spécifiques (SDL, MISRA C etc.)
- outils de modélisation : AADL, logique temporelle, réseaux de Petri,....

Logiciel embarqué = temps réel ?

- Les applications embarquées historiques étaient TR
- Les systèmes d'exploitation embarqués propriétaires sont TR (VxWorks, ...) $\rightarrow$ RTOS
- L'apparition des OS libres dans l'industrie et dans l'embarqué modifie la donne !
  - Linux est utilisable dans l'industrie
  - Linux n'est pas TR
  - Linux peut être modifié pour être TR (PREEMPT-RT, Xenomai)
  - Il existe des systèmes TR légers et libres (RTEMS, FreeRTOS,...)

Systèmes embarqués critiques : sécurité vs sûreté

Deux problèmatiques / domaines d'expertise distincts :

- sécurité informatique (ISO 27000 & co):
  - Disponibilité
  - Intégrité
  - Confidentialité
  - Auditabilité

  Attaquant $\rightarrow$ Bug $\rightarrow$ Intrusion

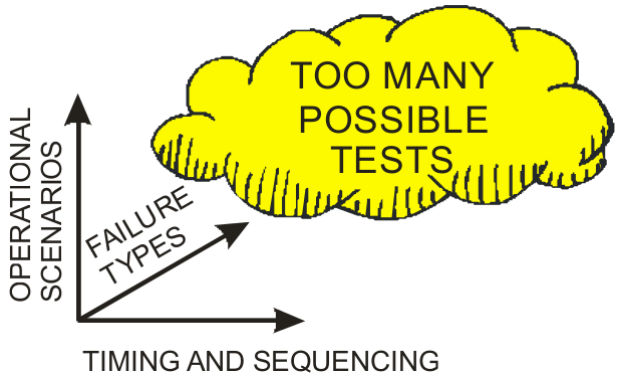
- sûreté de fonctionnement :
  - absense de conséquence pour l'utilisateur et l'environnement

  Faute $\rightarrow$ Erreur $\rightarrow$ Défaillance

Systèmes embarqués critiques : certification et normes

- La certification concerne
  - Tests fonctionnels et couverture de code
  - Bonnes Pratiques
  - Traçabilité
  - Documentation
- Exemples de normes pour la sûreté
  - DO-178C : aéronautique
    - 5 niveaux de criticité allant de « catastrophic » à « no effect »
  - ISO 26262 : automobile
    - ASIL Automotive System Integrity Level de S4 à S0
  - IEC 61508 : sureté de fonctionnement en général
    - SIL (System Integrity Level) de 4 à 1
  - IEC 60601 : médical

Systèmes embarqués critiques : tests ?



- Les tests sont utiles et importants
- Mais il est impossible de tout tester au niveau système
  - Trop de combinaisons de conditions opérationnelles, de séquences d'événements
  - Trop de types de défaillances, en partie intermittentes

Pour trouver une défaillance dont la probabilité d'occurence est de $10^{-8}/\text{h}$ il faut plus de 10^{8}h de tests.

$\Rightarrow$ Impossible de prouver la sécurité par les tests seuls.

Systèmes embarqués : architectures matérielles

Processeurs :

- micro-controleurs divers : monde industriel, hobbyistes (AVR / Arduino)
- ARM :
  - Cortex M (sans MMU), systèmes industriels,
  - Cortex A (MMU), Exynos, i.MX, Allwinnner,... : téléphones, tablettes, internet des objets,
- MIPS : équipements réseau, imprimantes,...
- SPARC : spatial (Leon)
- x86...

Bus dédiés :

- CAN : bus temps réel (contrôle de process, automobile,...)
- I2C, SPI, OneWire : liaisons séries simples avec E/S bas débit

Agenda

Introduction aux systèmes embarqués

Le logiciel libre dans l'embarqué

Outils de développement pour l'embarqué

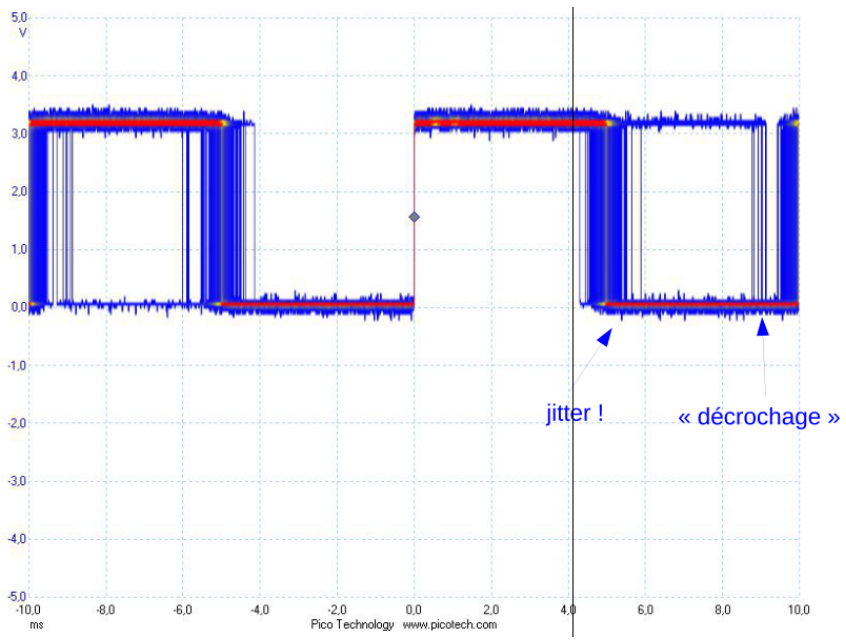
Importance du logiciel libre

- Le logiciel libre a pris une part importante des systèmes embarqués
  - Outils (GNU toolchain)
  - OS (Linux, RTEMS,...)
  - Ateliers de développement (Eclipse,...)
- La plupart des éditeurs commerciaux fournissent également (ou uniquement) des composants basés sur du logiciel libre.
- Problématique des licences...

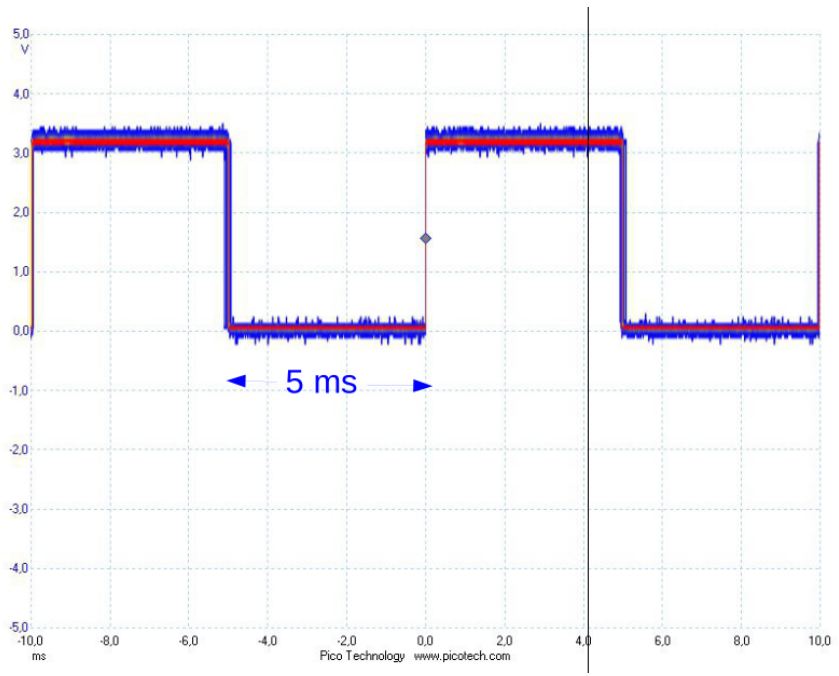
Linux comme (RT)OS

- Réservé aux systèmes *complexes*
  - 32 bits minimum
  - Gesion complex de la mémoire (MMU, pagination,...)
  - Empreinte mémoire importante : 2 Mo pour μCLinux, 4 Mo pour Linux
  - Consommation mémoire vive : 16 Mo minimum
- Nécessite des extensions (PREEMPT-RT ou Xenomai) pour temps-réel
- Incompatible avec les systèmes critiques
- Souvent utilisé pour les outils, simulateurs et dans des architectures « mixtes »
- Nombreux nouveaux domaines de l'embarqué (multimédia, réseau,...) où il convient parfaitement.

Sur un système chargé :



Noyau standard



Avec co-noyau Xenomai

Xenomai

- Xenomai est un sous-système temps-réel de Linux
  - Programmation de tâches en espace utilisateur,
  - API d'application et de pilotes temps-réel (RTDM) dédiés.
- intégré au noyau Linux $\rightarrow$ « Real-time sub-system »
- support de nombreuses architectures
- dispose de « Skins » permettant d'émuler des API temps-réel (POSIX, VxWorks, VRTX,...)
- Plus complexe à mettre en œuvre que PREEMPT-RT mais performances 5 à 10 fois supérieures
- licence GPL (cœur, LGPL (interfaces, espace utilisateur)

RTEMS

- RTEMS = **R**eal **T**ime **E**xecutive for **M**ultiprocessor **S**ystems
- initialement « Missile Systems » puis « Military Systems »
- Exécutif temps-réel embarqué diffusé sous licence libre (GPL avec exception)
- Pas exactement un système d'exploitation car mono-application (mais *multi-thread*)
- Programmation C, C++, Ada
- Plus de 100 BSP disponibles pour 20 architectures
- API RTEMS « classique » ou POSIX
- Utilisé par Airbus/EADS, ESA, NASA, NAVY,...

Android

- Système d'exploitation « libre » basé sur un noyau Linux modifié par Google
- Basé sur Dalvik, une machine virtuelle Java optimisée pour l'embarqué
- Graphique accéléré basé sur OpenGL ES
- Support multimédia (audio/video)
- Environnement de développement intégré basé sur Eclipse (Émulateur, déboguer) $\rightarrow$ ADT

Utilisable pour des projets :

- avec IHM
- sans TR dur

Agenda

Introduction aux systèmes embarqués

Le logiciel libre dans l'embarqué

Outils de développement pour l'embarqué

Développement croisé

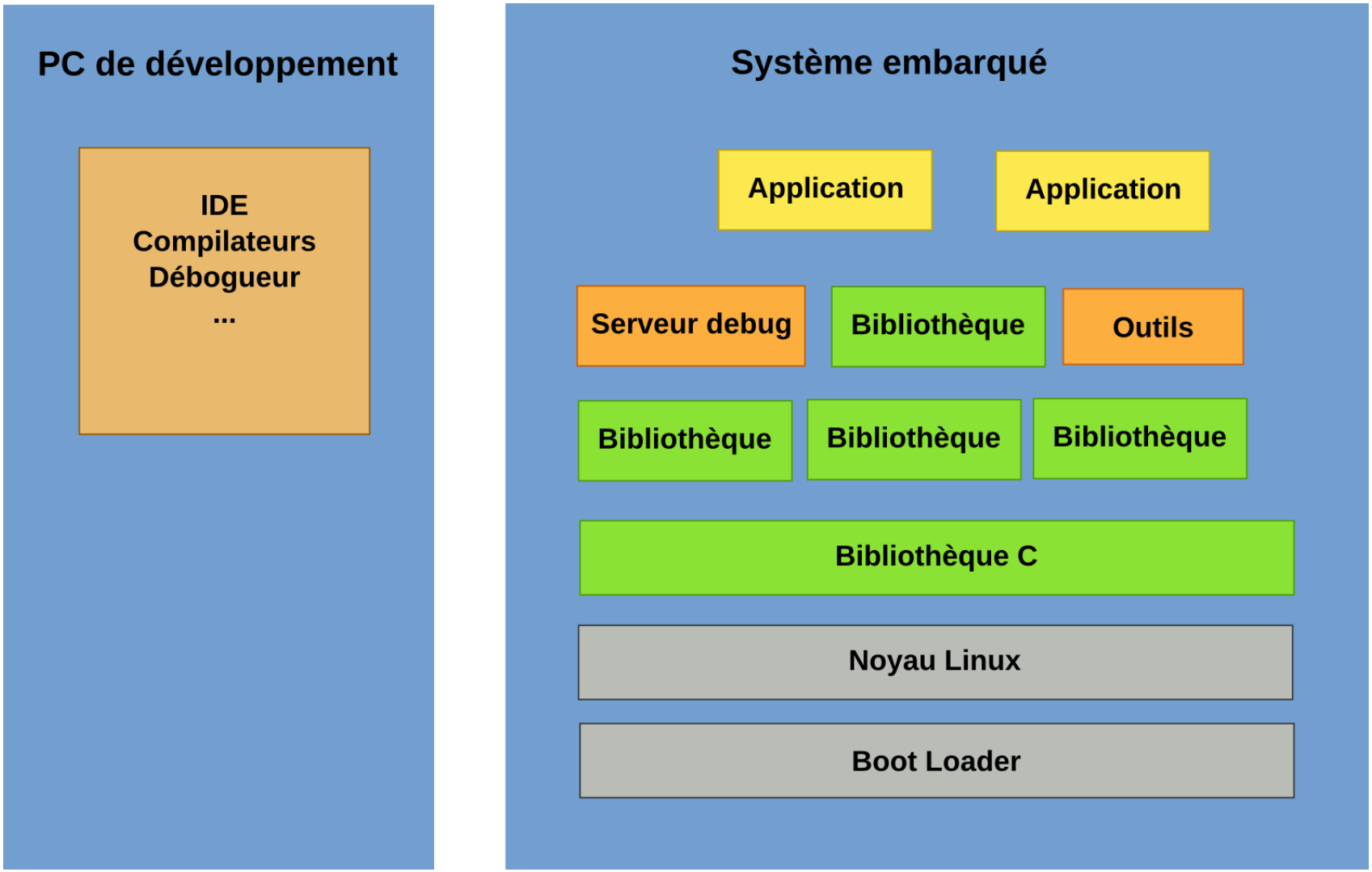
Définition :

- Machine hôte : qui exécute les outils de développement
- Machine cible : qui exécute le système et/ou les applications embarquées

Outils nécessaires :

- compilateur, éditeur de liens,
- debogueur, simulateur
- EDI/IDE (Eclipse)
- sondes JTAG
- outils de fabrication de paquets et d'images disques bootables
- gestion de version

Architecture type



Production de distribution

C'est un outil qui :

- crée la distribution à partir des sources
  des composants adaptés en applicant des « patches »
  - Il ne s'agit pas de *distribution* mais d'outil de *création de distribution*
- ne fournit pas les sources mais les *règles de production* et prend en compte les dépendances
- peut produire la chaîne de compilation croisée (gcc)
- produit les différents éléments de la distribution
  - image du bootloader
  - noyau Linux
  - image du système de fichiers

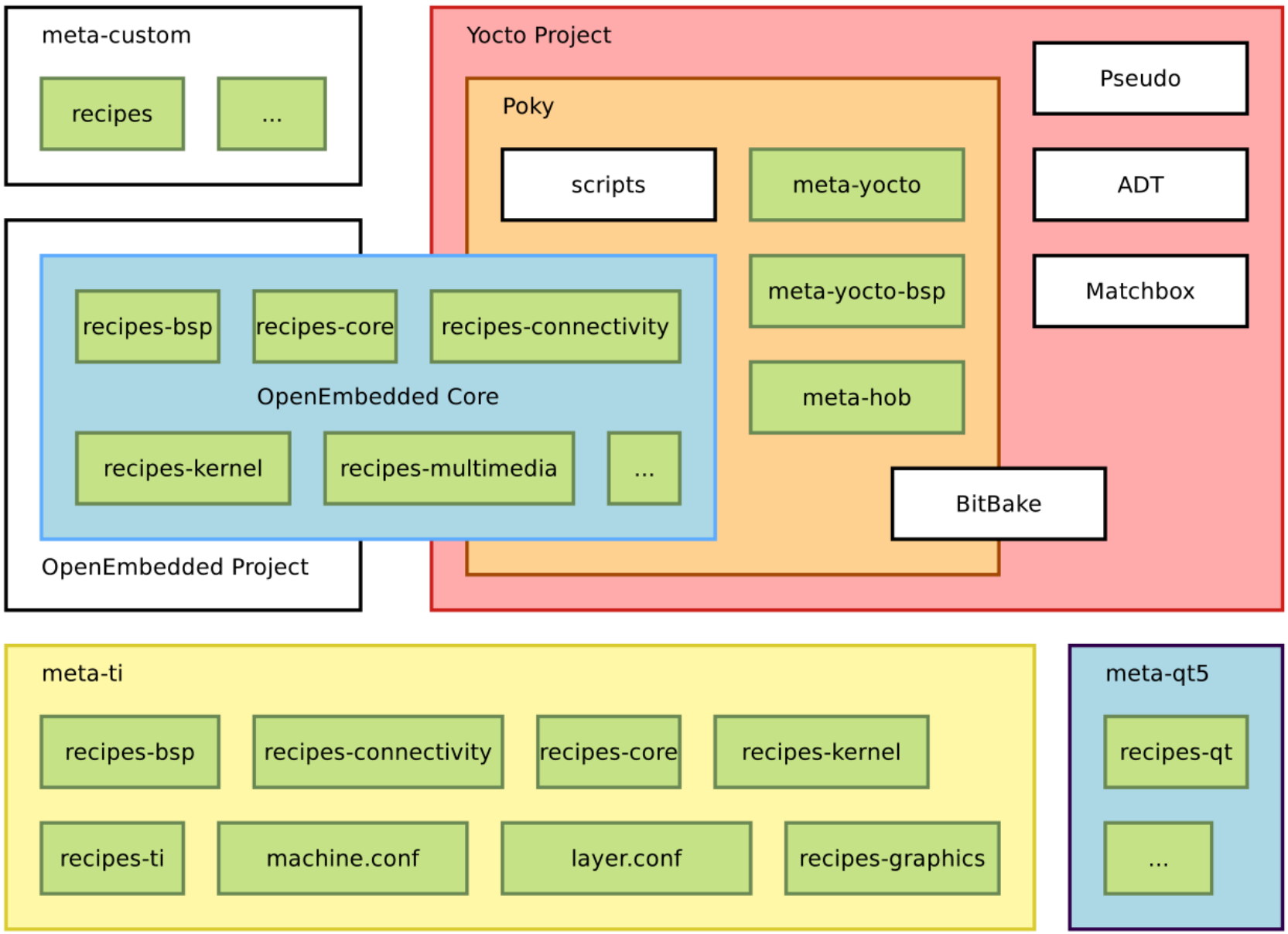
Les principaux outils disponibles

- Yocto / OpenEmbedded
  - Moteur écrit en python
  - Très puissant mais très lourd
  - Basé sur des « recettes » (outil bitbake)

- Buildroot
  - Basé sur la commande make
  - Moins riche que OE, mais plus simple et plus rapide
  - Bien Maintenu

- OpenWRT
  - dérivé de BuildRoot
  - orienté vers les IAD (Internet Access Device)

Yocto


meta-custom
recipes
...
Yocto Project
Pseudo
Poky
scripts
meta-yocto
ADT
meta-yocto-bsp
Matchbox
recipes-bsp
recipes-core
recipes-connectivity
OpenEmbedded Core
meta-hob
recipes-kernel
recipes-multimedia
...
BitBake
OpenEmbedded Project
meta-ti
recipes-bsp
recipes-connectivity
recipes-core
recipes-kernel
recipes-ti
machine.conf
layer.conf
recipes-graphics
meta-qt5
recipes-qt
...

Émulateur QEMU

- Émulateur de matériel développé par Fabrice Bellard, diffusé sous GPLv2
- Exécuté dans l'espace utilisateur de Linux
- Permet d'émuler diverses architectures : x86, PowerPC, ARM, etc.
- Émulation de carte complète $\rightarrow$ outil de développement, mise au point, test automatique.
  - outil de certification DO-178 (Couverture)
- Large communauté. `http://wiki.qemu.org/`

Autres Outils libres pour l'embarqué

Systèmes critiques :

- OCARINA (TPT) : « compilateur » AADL (Architecture Analysis and Design language)
- SynDEx (INRIA) : Synchronized Distributed Executive, générateur de code « synchrone »
- POK (TPT) : RTOS ARINC 653
- TOPCASED (Airbus) : Toolkit in OPen source for Critical Applications & SystEms Development
- Initiative Open-DO Adacore

Robotique :

- Ecosystème ROS de la Open Source Robotics Foundation
- Orocos Open Robot Control Software
- Bride Plugin eclipse pour développement ROS et Orocos

Conclusion

- L'embarqué est un marché en croissance
- Augmentation significative de l'activité « embarqué » chez les SSII et les éditeurs
- Les marchés traditionnels restent leaders
  - aéronautique/spatial/défence (62%)
  - automobile (47%)
- Augmentation de l'utilisation du logiciel libre (outils)
  - Éditeurs (65 %)
  - SSII (42%)
- Source : étude Syntec 2012

Questions ?

Références

- Portail Linux embarqué : eLinux.org
- Linux Embarqué, Pierre Ficheux, éditions Eyrolles, 2012.
- Logiciel Libre et sûreté de fonctionnement, sous la direction de Philippe David et Hélène Waeselynck, éditions Hermes Science, 2003 (pdf).
- Better Embedded System Software, Philipp Koopman, Drumnadrochit Education LLC, 2010.
- A case study of Toyota unintented acceleration and software study, Philip Koopman, Carnegie Mellon University, 2014.
- Supports de Formation de Free Electrons.