www.ciose.fr


**C. Charreyre**
christian.charreyre@ciose.fr
http://www.ciose.fr
@CIOinfoindus


# Concevoir un système embarqué Linux avec YOCTO Project

www.ciose.fr

# Licence

## Attribution-Noncommercial-Share Alike 4.0 International

- You are free:
  - to Share - copy and redistribute the material in any medium or format
  - to Adapt - remix, transform, and build upon the material

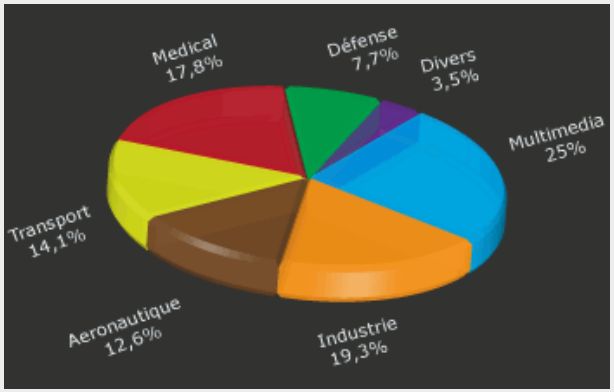
  The licensor cannot revoke these freedoms as long as you follow the license terms.
- Under the following conditions:
  - Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - NonCommercial — You may not use the material for commercial purposes.
  - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- License text : http://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

CAP'TRONIC

www.ciose.fr

# CIO en quelques mots

- Société d'ingénierie en systèmes embarqués : électronique et logiciel
- Au service de nos clients depuis 1990
- Une équipe de 15 spécialistes en embarqué et temps réel
- Expert en Linux embarqué depuis 2000, OpenEmbedded et Yocto depuis 2008
- CA annuel : 1 500 000 €
- Siège à Saint-Etienne
- Agence à Marseille
- Agréé CIR




CAP'TRONIC

www.ciose.fr

# Contexte technique et business

CAP'TRONIC

www.ciose.fr

# Les tendances en Linux embarqué

- 10 ans en arrière, Linux = alternative aux RTOS propriétaires → peu de besoins fonctionnels :
  - Kernel et glibc
  - Fabrication d'un Root File System autour de Busybox
  - Application métier développée en interne
  - Peu de problèmes de complexité et de cohérence
  - Nécessité d'une bonne connaissance de Linux pour créer sa distribution embarquée
- Linux = un OS parmi d'autres, pas de développement massif

CAP'TRONIC

www.ciose.fr

# Les tendances en Linux embarqué

- Aujourd'hui: forte pression pour développer des smart devices dans des Time To Market réduits
  - Moins (plus?) de limitations de mémoire vive ou de masse
  - Processeurs beaucoup plus puissants
  - IHM riches avec audio, vidéo, graphisme évolué (impact iOS et Androïd)
  - Besoins importants de connectivité : Wifi, Bluetooth, 3G
  - Time to Market de plus en plus court
  - Recentrage sur la valeur ajoutée

CAP'TRONIC

www.ciose.fr

# Les tendances en Linux embarqué

- Compétition sur la spécificité – coopération sur les environnements de base
  - Ex : Initiative Genivi dans le monde automobile (In Vehicle Infotainment) – Création d'une distribution basée Yocto

CAP'TRONIC

www.ciose.fr

# Quelques exemples

- Devices bâtis sur base Embedded Linux :






CAP'TRONIC

# Le couple fort : ARM + Linux

- Beaucoup de processeurs ARM au coté des architectures X86 :
  - Fondeurs multiples
  - Intégration processeur + fonctions micro contrôleur sur un SOC
  - Faible consommation (relatif)
  - Rapport coût / performance souvent intéressant
- Solutions ARM souvent accompagnées d'OS Linux

# Le challenger Androïd

- Forte demande Androïd sur le marché de l'embarqué :
  - Diffusion de l'OS auprès des utilisateurs par smartphone / tablettes
  - Attrait d'une interface unique domaine personnel / domaine professionnel
  - Importance de la connectivité
  - Attrait du Play Store pour les smart devices
  - Sentiment Androïd = une version de Linux

CAP'TRONIC

www.ciose.fr

# Le challenger Androïd

- Androïd = OS différent, avec un Linux modifié « sous le capot »
- Le monde Linux reste totalement caché aux développeurs
  - Développement normal en Java au dessus du framework Androïd
  - Paradigmes de développement très éloignés des habitudes de l'embarqué
- Une version Open Source AOSP, et une version interne Google
- Seul Google pilote l'évolution d'Androïd

CAP'TRONIC

www.ciose.fr

# Le challenger Android

- Nécessité de réussir les tests de compatibilité CTS de Google pour avoir accès
  - Aux applications spécifiques Google
  - Au store Play Store
  - Prérequis matériels et logiciels
- Nécessité de porter Android sur sa plateforme
  - Plus complexe qu'un portage Linux
    - Plus de documentation si pas enregistré auprès de Google
  - Drivers au niveau Linux + HAL
  - Kernel Linux spécifique ⇒ patches nécessaires sur kernel Linux déjà dispo sur la plateforme
- Bien réfléchir au choix Androïd vs Linux

CAP'TRONIC

# Pourquoi un outil de build ?

www.ciose.fr

# Comment répondre au challenge Linux embarqué ?

- Utiliser des éléments logiciels issus du riche écosystème Linux
- Les assembler comme des briques modulaires, tels quels ou adaptés
- Développer seulement les éléments applicatifs non disponibles sur étagère :
  - ⇒ Mouvement d'une culture de développeur vers une culture d'assembleur

CAP'TRONIC

www.ciose.fr

# Comment répondre au challenge Linux embarqué ?

- Nécessité d'outils adaptés pour gérer la fragmentation et la complexité de l'écosystème Linux, et assurer une cohérence globale :
  - Outils de génération de distributions Linux embarquée

CAP'TRONIC

www.ciose.fr

# Travailler avec Linux

- Un monde fragmenté aux multiples sources
  - Bootloaders (UBoot, RedBoot, LILO, Grub, ...)
  - Kernel (kernel.org, fournisseur hardware, ...)
  - Librairies de base (glibc ou alternatives réduites)
  - Bases applicatives (busybox, kits embarqués libres ou propriétaires, ....)
  - IHM (Qt, MicroWindows/NanoX, ...)
  - Multimédia (Mplayer, Gstreamer, Xine, ....)
  - Extensions temps réel (RTAI, Xenomai, …)
- Qu'il faut assembler en un ensemble cohérent : votre device
- Mais chaque projet contributeur vit sa vie à son propre rythme : cycle de vie propre, sans gestion centralisée

CAP'TRONIC

# Travailler avec Linux

- Faire attention au respect de licences multiples (GPL, LGPL, BSD, etc...)
  - Les connaître et les respecter
  - Adapter ce que l'on utilise à sa stratégie de publication des codes source

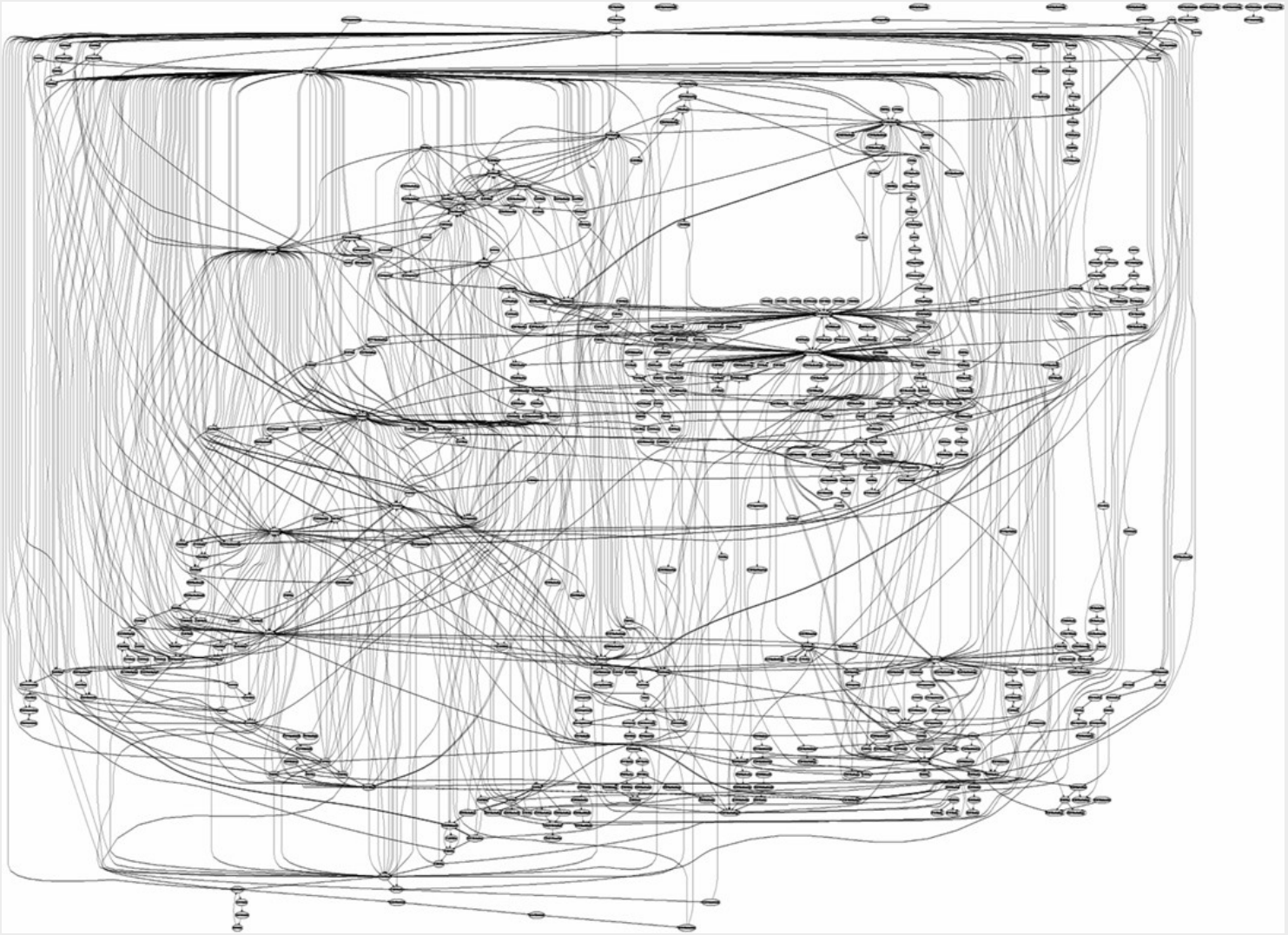
# Pourquoi un outil de build ?

- Pour éviter cela

www.ciose.fr

# Pourquoi un outil de build ?

- Pour gérer cela



CAP'TRONIC

# Pourquoi un outil de build ?

- Dépendances de nautilus (file manager) : 62 librairies

```
cch@cch-tosh: ~
Fichier  Édition  Affichage  Terminal  Aide

cch@cch-tosh:~$ ldd /usr/bin/nautilus
        linux-vdso.so.1 =>  (0x00007fff563a1000)
        libSM.so.6 => /usr/lib/libSM.so.6 (0x00007f75394b6000)
        libICE.so.6 => /usr/lib/libICE.so.6 (0x00007f753929b000)
        libXrender.so.1 => /usr/lib/libXrender.so.1 (0x00007f7539090000)
        libnautilus-extension.so.1 => /usr/lib/libnautilus-extension.so.1 (0x00007f7538e86000)
        libappindicator.so.0 => /usr/lib/libappindicator.so.0 (0x00007f7538c7d000)
        libgnome-desktop-2.so.17 => /usr/lib/libgnome-desktop-2.so.17 (0x00007f7538a52000)
        liblaunchpad-integration.so.1 => /usr/lib/liblaunchpad-integration.so.1 (0x00007f753884e000)
        libunique-1.0.so.0 => /usr/lib/libunique-1.0.so.0 (0x00007f7538641000)
        libdbus-glib-1.so.2 => /usr/lib/libdbus-glib-1.so.2 (0x00007f753841e000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00007f7538201000)
        libgailutil.so.18 => /usr/lib/libgailutil.so.18 (0x00007f7537ff9000)
        libgtk-x11-2.0.so.0 => /usr/lib/libgtk-x11-2.0.so.0 (0x00007f75379d6000)
        libgdk-x11-2.0.so.0 => /usr/lib/libgdk-x11-2.0.so.0 (0x00007f7537729000)
        libatk-1.0.so.0 => /usr/lib/libatk-1.0.so.0 (0x00007f7537508000)
        libgio-2.0.so.0 => /usr/lib/libgio-2.0.so.0 (0x00007f7537254000)
        libgdk_pixbuf-2.0.so.0 => /usr/lib/libgdk_pixbuf-2.0.so.0 (0x00007f7537038000)
        libcairo.so.2 => /usr/lib/libcairo.so.2 (0x00007f7536db5000)
        libpango-1.0.so.0 => /usr/lib/libpango-1.0.so.0 (0x00007f7536b6a000)
        libgobject-2.0.so.0 => /usr/lib/libgobject-2.0.so.0 (0x00007f7536922000)
        libgmodule-2.0.so.0 => /usr/lib/libgmodule-2.0.so.0 (0x00007f753671e000)
        libgthread-2.0.so.0 => /usr/lib/libgthread-2.0.so.0 (0x00007f7536518000)
        libgconf-2.so.4 => /usr/lib/libgconf-2.so.4 (0x00007f75362db000)
        libglib-2.0.so.0 => /lib/libglib-2.0.so.0 (0x00007f7535ffd000)
        libxml2.so.2 => /usr/lib/libxml2.so.2 (0x00007f7535cac000)
        libX11.so.6 => /usr/lib/libX11.so.6 (0x00007f7535976000)
        libexif.so.12 => /usr/lib/libexif.so.12 (0x00007f7535731000)
        libexempi.so.3 => /usr/lib/libexempi.so.3 (0x00007f7535444000)
        libselinux.so.1 => /lib/libselinux.so.1 (0x00007f7535226000)
        libm.so.6 => /lib/libm.so.6 (0x00007f7534fa3000)
        libc.so.6 => /lib/libc.so.6 (0x00007f7534c1f000)
        libuuid.so.1 => /lib/libuuid.so.1 (0x00007f7534a1a000)
        libindicator.so.0 => /usr/lib/libindicator.so.0 (0x00007f753480f000)
        libpangoft2-1.0.so.0 => /usr/lib/libpangoft2-1.0.so.0 (0x00007f75345e4000)
        libpangocairo-1.0.so.0 => /usr/lib/libpangocairo-1.0.so.0 (0x00007f75343d7000)
        libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x00007f7534151000)
        libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0x00007f7533f1b000)
```

CAP'TRONIC

www.ciose.fr

# Des images reproductibles

- Nécessité de maîtriser la production des images embarquées
  - Dans un contexte riche avec de multiples dépendances
  - Dans un contexte évolutif (dynamique logiciel libre)
  - Dans la durée (pérennité des produits)
  - A l'identique ou avec des corrections / évolutions
- Solutions manuelles à bannir
  - Dépendantes des acteurs (manque de formalisme, le gourou parti ou absent comment fait on?)
  - Atteignent leurs limites face à l'inflation des composants à gérer

CAP'TRONIC

www.ciose.fr

# Des images reproductibles

- Robustesse de la production aux changements d'infrastructure de développement
  - Changement de matériel ou de version d'OS : quel impact ?
  - Risque réduit par utilisation de Virtual Machines
- Garantir la possibilité de rebâtir l'image identique, mais également de la faire évoluer de façon maîtrisée
  - Reproductible ≠ Figé
- Intégrer tous les paramétrages à la production de l'image (logins, réseau, langue, etc....)
  - Suppression des réglages manuels opérés lors de l'installation

CAP'TRONIC

www.ciose.fr

# Isolation poste développeur / cible

- Le Root File System est d'abord construit sur le poste de développeur
- Les logiciels embarqués sont cross compilés sur le poste développeur
- Nécessité d'éviter tout mélange entre hôte et cible :
  - Processus de compilation : headers et libraires croisées vs natives
  - Processus d'installation : librairies, exécutables et fichiers de contrôle
- Attention aux mélanges !!!
- Problème général mais amplifié dans le cas Linux

CAP'TRONIC

www.ciose.fr

# Isolation poste développeur / cible




CAP'TRONIC

www.ciose.fr

# Isolation poste développeur / cible




# Les outils disponibles

- Des besoins communs :
  - Sélectionner les composants logiciels
  - Configurer
  - Cross Compiler
  - Installer dans le File System image (en cours de construction)
  - Conditionner le File System au format final
- Besoins liés à la logique de développement Linux :
  - ./configure – make – make install
- Dans un contexte de développement croisé :
  - Complication liée à la séparation hôte - cible

CAP'TRONIC

# Les outils disponibles

- Tour d'horizon :
  - Do It Yourself : limité aux cas simples
    - Avantage = maîtrise totale
    - Inconvénient = il faut tout faire
  - Buildroot : simple mais fonctionnellement moins riches que les autres
    - Adapté aux applications enfouies, pas très riches
    - Difficile de travailler en différentiel : régénération complète du File System, pas de gestion de paquets
    - Basé sur des Makefiles
  - Scratchbox : riche mais obsolète
  - LTIB : outil utilisé par Freescale, mais changement en cours au profit du Yocto Project
    - Versions logicielles datées (host + target)

CAP'TRONIC

# Les outils disponibles

- Tour d'horizon :
  - OpenEmbedded
    - Ancêtre commun issu du projet Open Zaurus, toujours actif. Base de distributions variées
  - Angström
    - Distribution orientée machines ARM, construite à partir de OpenEmbedded core.
  - Yocto Project
    - Projet pour mettre en place de manière industrielle des outils de création de distribution Linux embarqué
    - Sous l'égide de la Linux Foundation
    - Implémente la distribution Poky en se basant sur Open Embedded
    - Par abus de langage dans la suite du document, nous utilisons Yocto pour l'ensemble des concepts évoqués.
  - OpenEmbedded / Angström / Yocto (Poky) cousins

CAP'TRONIC

www.ciose.fr

# Les outils disponibles

- OpenEmbedded Core fondement de Yocto et Angström
- Même syntaxe, même moteur :
  - Assemblage possible entre recettes issues des 3 outils
  - Tout ce qui est vrai pour Yocto est valable pour Angström

CAP'TRONIC

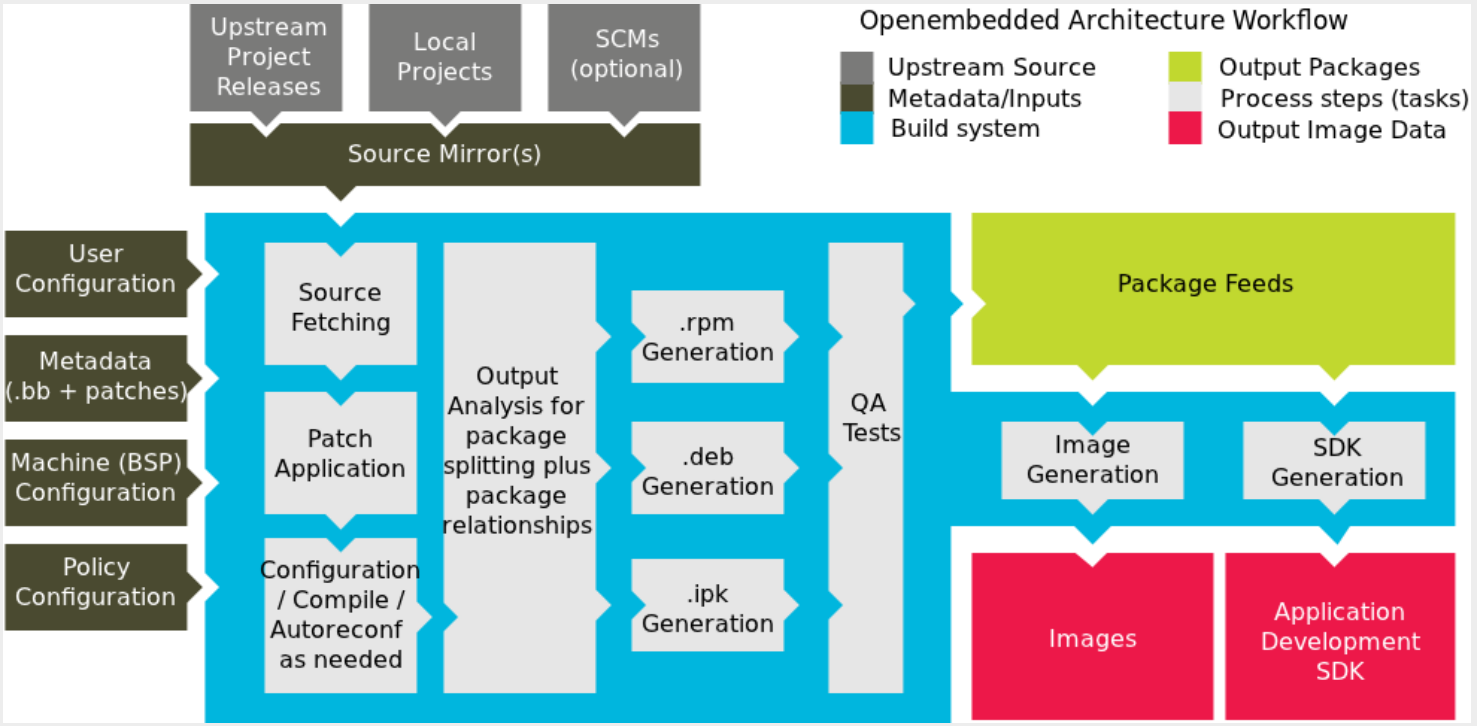
www.ciose.fr

# Utilisation de Yocto

CAP'TRONIC

www.ciose.fr

# Workflow de Yocto




**Credit – Yocto Project**

CAP'TRONIC

# Workflow de Yocto

- Les entrées :
  - Code source des projets « upstream » :
    - Fetchés depuis le site projet ou depuis un miroir Yocto
    - Formats variés : archive, révision dans un SCM ...
    - Protocoles variés : http(s), ftp, svn, git …
  - Code source local (projet interne) :
    - Archive locale ou accès à un serveur SCM interne
  - Fichiers de configuration :
    - Caractéristiques de la machine cible (kernel, bootloader, format image, tuning compilateur ...)
    - Caractéristiques de la distribution (paquets inclus, versions, choix entre alternatives, choix libc …)
    - Caractéristiques des layers (détaillé ultérieurement)
    - Configuration du build : machine, distribution, layers actives, format paquet ...

CAP'TRONIC

# Workflow de Yocto

- Le cœur :
  - Un moteur d'exécution de tâches écrit en Python : bitbake
  - Exécute automatiquement les tâches nécessaires à la fabrication de la cible fournie
  - Fonctionne en ligne de commande
  - Exemple : `bitbake core-image-minimal`

# Workflow de Yocto

- Les sorties :
  - Les paquets binaires des composants utilisés (formats ipk, deb ou rpm)
    - Possibilité d'embarquer un gestionnaire de paquets sur la cible
    - Permet de travailler en différentiel pour mise à jour / enrichissement
  - L'image finale déployable sur la cible (formats variés : tar.bz2, ext3, ubi etc....)
  - Un SDK exportable pour les développeurs d'application
  - Un récapitulatif des paquets embarqués et de leur licences

CAP'TRONIC

www.ciose.fr

# Le moteur bitbake

- Un moteur écrit en Python : bitbake
- Un jeu de recettes pour fabriquer les paquets logiciels
- Une notion de classes pour mise en commun entre recettes
- Une notion de méta paquets pour structurer (packagegroup)
- Des dépendances entre paquets, décrites dans les recettes, ou déterminées automatiquement (librairies partagées)
- Pour chaque recette des tâches élémentaires
- Calcul de l'arbre des dépendances pour fabriquer les paquets dans le bon ordre

CAP'TRONIC

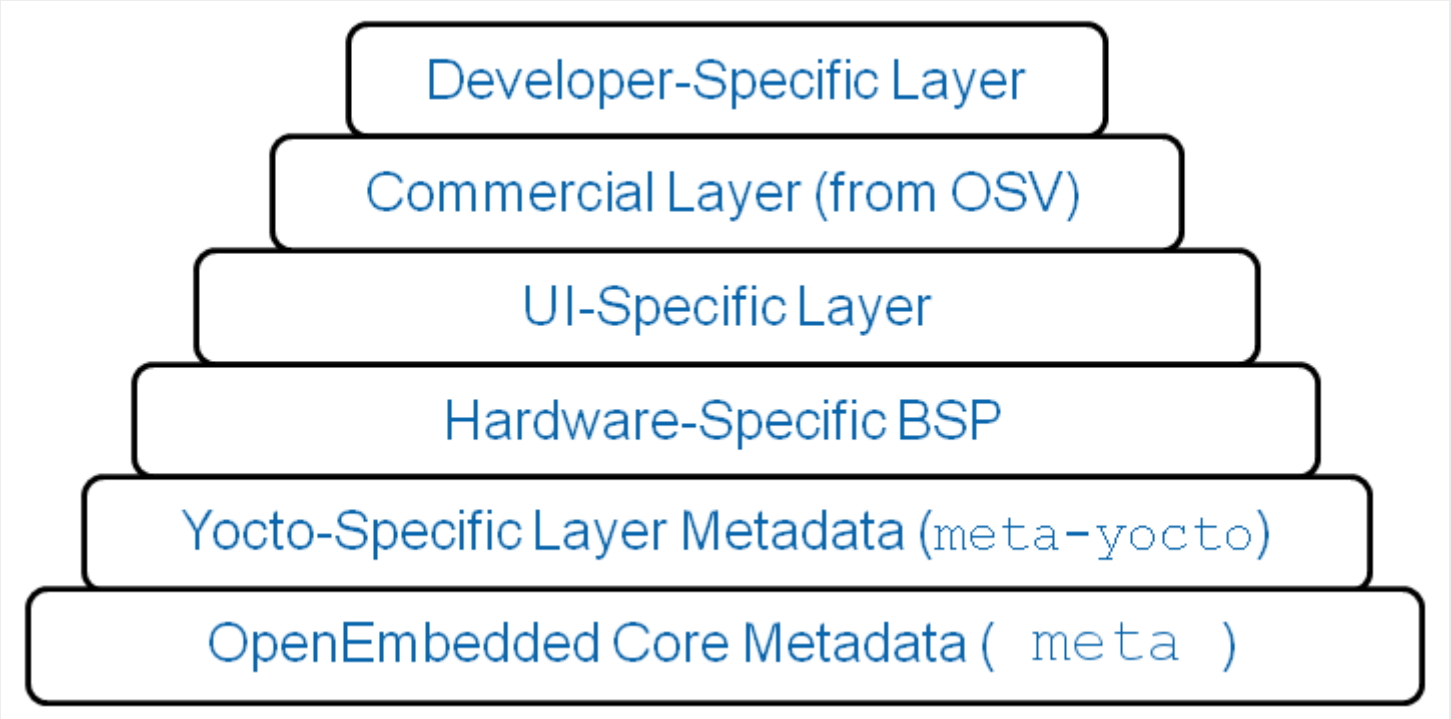
# Principales tâches élémentaires

| Tâche | Rôle |
|---|---|
| fetch | Téléchargement depuis dépôt upstream |
| unpack | Extraction dans répertoire travail |
| patch | Application de patches issus des recettes |
| configure | Configuration |
| compile | Compilation croisée |
| install | Installation dans tampon local au composant |
| populate_lic | Installation fichier(s) licence dans répertoire deploy |
| package | Fabrication des descriptifs des packages |
| populate_sysroot | Déploiement des paquets dans le sysroot |
| package_write_xxx | Création du paquet au format xxx |

CAP'TRONIC

# Structuration en couches

- Architecture en couches
- Priorité entre couches qui partageraient une recette



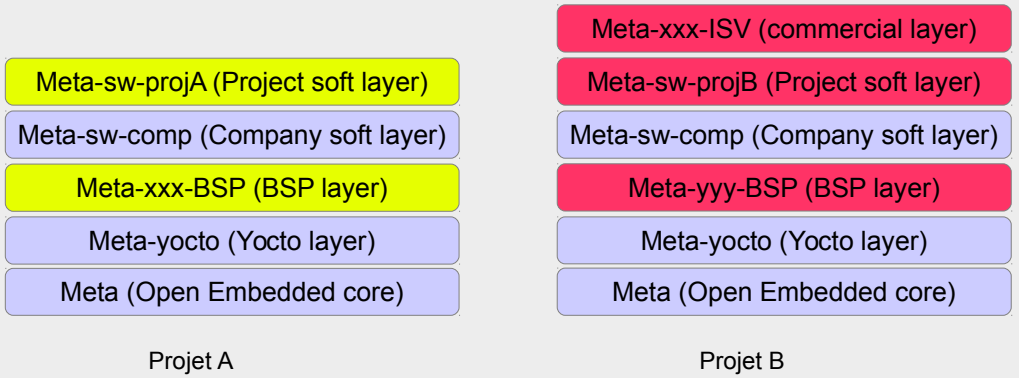

Liste des couches : http://layers.openembedded.org/layerindex/layers/

# Structuration en couches

- Possibilité de créer sa propre layer (niveau société)
- Possibilité de créer des layers par affaire ou projet
- Le but est d'optimiser la réutilisation des recettes en évitant au maximum la duplication
- Si une même recette présente dans plusieurs layers, c'est la layer de priorité supérieure qui s'impose.

# Structuration en couches

- L'utilisation des couches améliore la réutilisabilité




- Les recettes peuvent être modifiées dans les couches supérieures à l'aide d'une sorte de patch (fichier bbappend) :
  - Recette originale inchangée dans la couche inférieure
  - Le fichier bbappend personnalise la recette dans la couche supérieure

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette

- Une recette pour un ou plusieurs paquets (fractionnement pour optimisation de l'espace)
- Des variables d'environnement
- Des tâches élémentaires implicites ou explicites (pour modifier l'implicite) : langage shell et python
- Jeu de recettes géré par la communauté Yocto → solution au problème de complexité :
  - Cohérence entre versions de composants dépendants
  - Mise à jour des recettes fonction des évolutions upstream
  - Application de patches locaux quand cela est nécessaire

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files, syslog and other
destinations"

SECTION = "libs"

HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING;md5=7fbc338309ac38fefcd64b04bb903e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/${PN}/log4c-${PV}.tar.gz \

file://oe.patch \

file://add-pkgconfig-data.patch \

"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"

SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8ab1e442b"
```

Descriptif

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files, syslog and other
destinations"

SECTION = "libs"

HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING;md5=7fbc338309ac38fefcd64b04bb903e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/${PN}/log4c-${PV}.tar.gz \

file://oe.patch \

file://add-pkgconfig-data.patch \

"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"

SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8ab1e442b"
```

Licence

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files, syslog and other
destinations"

SECTION = "libs"

HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING;md5=7fbc33                    03e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/${PN}/log4c-${PV}.tar.gz \

file://oe.patch \

file://add-pkgconfig-data.patch \

"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"

SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8ab1e442b"
```

Révision recette

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files, syslog and other
destinations"

SECTION = "libs"

HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING;md5=7fbc338309ac38fefcd64b04bb903e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/${PN}/log4c-${PV}.tar.gz \

file://oe.patch \

file://add-pkgconfig-data.patch \

"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"

SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8ab1e442b"
```

Sources + patches locaux

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files, syslog and other
destinations"

SECTION = "libs"

HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING;md5=7fbc338309ac38fefcd64b04bb903e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/${PN}/log4c-${PV}.tar.gz \

file://oe.patch \

file://add-pkgconfig-data.patch \

"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"

SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8ab1e442b"
```

Classes héritées

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files, syslog and other
destinations"

SECTION = "libs"

HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"

LIC_FILES_CHKSUM = "file://COPYING;md5=7fbc338309ac38fefcd64b04bb903e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/${PN}/log4c-${PV}.tar.gz \

file://oe.patch \

file://add-pkgconfig-data.patch \

"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"

SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8ab1e442b"
```

Checksums

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (ed)

```
DESCRIPTION = "a line-oriented text editor"
HOMEPAGE = "http://www.gnu.org/software/ed/"
BUGTRACKER = ""

LICENSE = "GPLv3+"
LIC_FILES_CHKSUM = "file://COPYING;md5=f27defe1e96c2e1ecd4e0c9be8967949 \

file://ed.h;endline=20;md5=c708cda1b2e8d723d458690b7db03878 \

file://main.c;endline=24;md5=1bd039d59e04ee5f82adcc970144a2c3"

SECTION = "base"
PR = "r0"

# LSB states that ed should be in /bin/
bindir = "${base_bindir}"

SRC_URI = "${GNU_MIRROR}/ed/ed-${PV}.tar.gz \
           file://ed-1.2-build.patch"

SRC_URI[md5sum] = "9a78593decccaa889523aa4bb555ed4b"
SRC_URI[sha256sum] =
"211c67b0c4aae277d34b1c5f842db1952e468e5905142868e4718ac838f08a65"

do_configure() {
     ${S}/configure
}

do_install() {
     oe_runmake 'DESTDIR=${D}' install
}
```

Tâches explicites

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (gthumb)

```
DESCRIPTION = "gThumb is an image viewer and browser for the GNOME Desktop"
SECTION = "x11/gnome"
LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=59530bdf33659b29e73d4adb9f9f6552"
DEPENDS = "glib-2.0 gtk+ libxml2 gnome-doc-utils libunique gconf libpng
gstreamer jpeg tiff gst-plugins-base"

PR = "r4"

EXTRA_OECONF = "--disable-gnome-keyring --disable-libso
--disable-clutter"

inherit gnome pkgconfig

SRC_URI[archive.md5sum] = "97fc13221b0c5d80c27a2e25a3a3ac6f"
SRC_URI[archive.sha256sum] =
"cf809695230ab8892a078be454a42ade865754c72ec1da7c3d74d4310de54f1d"

SRC_URI += "file://parallel.patch"

do_install_append () {
     rm ${D}${libdir}/${BPN}/extensions/*.a
}

FILES_${PN} += "${datadir}/icons"
FILES_${PN} += "${libdir}/${BPN}/extensions/*.so \
                ${libdir}/${BPN}/extensions/*.extension"
FILES_${PN}-dev += "${libdir}/${BPN}/extensions/*.la"
FILES_${PN}-dbg += "${libdir}/${BPN}/extensions/.debug/"
```

Dépendances

Directives packaging

CAP'TRONIC

www.ciose.fr

# Anatomie d'une recette (image)

```
include recipes-sato/images/core-image-sato.bb
IMAGE_FEATURES += "debug-tweaks"
DISTRO_FEATURES += "pulseaudio"
WEB = "web-webkit"

# Add extra image features
EXTRA_IMAGE_FEATURES += " \
    ${SOC_EXTRA_IMAGE_FEATURES} \
    nfs-server \
    tools-debug \
    tools-profile \
    qt4-pkgs \
"

IMAGE_INSTALL += " \
    ${SOC_IMAGE_INSTALL} \
    cpufrequtils \
    nano \
    packagegroup-fsl-gstreamer \
    packagegroup-fsl-tools-testapps \
    packagegroup-fsl-tools-benchmark \
    packagegroup-qt-in-use-demos \
    qt4-plugin-phonon-backend-gstreamer \
    qt4-demos \
    qt4-examples \
    fsl-gui-extrafiles \
    "

export IMAGE_BASENAME = "fsl-image-gui"
```

Sélection de features

Sélection de packagroups et composants

CAP'TRONIC

# Modification d'une recette en delta

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${P}:"

dirs755 += " ${localstatedir}/volatile/mqueue"

volatiles += "mqueue"

SRC_URI += "file://root-profile"

CONFFILES_${PN} += "/home/root/.profile"

do_install_append() {

install -m 0755 ${WORKDIR}/root-profile $
{D}/home/root/.profile

}
```

CAP'TRONIC

# Création d'une image

- Une commande unique fabrique l'image déployable :
  - `bitbake qt4e-demo-image`
  - Calcul de toutes les dépendances
  - Enchaînement automatique de toutes les tâches élémentaires, jusqu'à l'image finale

# Création d'un paquet individuel

- Une commande unique fabrique le paquet déployable :
  - `bitbake busybox`
  - Calcul de toutes les dépendances
  - Enchaînement automatique de toutes les tâches élémentaires, jusqu'au(x) paquet(s) final(aux)

www.ciose.fr

# Création de recettes nouvelles

- Nombreuses recettes disponibles
  - Utiliser l'existant (voir layers index)
- Nécessité de créer ses propres recettes pour :
  - Logiciel existant mais pas de recette disponible
  - Logiciels développés in house
  - Images et packagegroup propres
- Définir les diverses variables nécessaires
- Définir explicitement les diverses tâches
  - Usage des autotools facilite les choses : classe dédiée définit automatiquement toutes les tâches

CAP'TRONIC

www.ciose.fr

# Adaptation de recettes existantes

- Créer des fichiers bbappend dans une couche de niveau supérieur
- Permet de personnaliser les réglages
- Permet d'appliquer des patches ou de modifier le packaging
- Etc....

CAP'TRONIC

# Gestion du versionning hôte

- 2 niveaux de versionning :
  - Version du logiciel (gérée par l'équipe projet qui développe ce logiciel upstream)
  - Version de la recette (gérée par la communauté Yocto)
- Plusieurs recettes possibles pour un même logiciel (différentes versions du logiciel + svn/git)
- Par défaut version la + élevée retenue – peut être contré par paramétrage au niveau distribution

CAP'TRONIC

www.ciose.fr

# Gestion du versionning cible

- Gestionnaire de paquets sur la cible :
  - Installation
  - Suppression
  - Upgrade
- Gère les dépendances à l'installation - suppression
- Gère les versions du logiciel + version de la recette :
  - Refus des downgrade sauf forçage

CAP'TRONIC

www.ciose.fr

# Création d'un BSP pour un nouveau hardware

- Créer une couche dédiée au hardware
- Créer les recettes des spécificités liées au matériel :
  - Bootloader
  - Kernel
  - Serveur Xorg
  - Éléments de facteur de forme
  - Éventuellement codecs, librairies graphiques accélérées etc...
- Définir les caractéristiques propres au hardware
  - Architecture
  - Tuning cross compilateur ….

CAP'TRONIC

# Reproductibilité et isolation

- Yocto permet de :
  - Supprimer toute tâche manuelle : `bitbake my-image`
  - Définir tous les paramétrages de l'image à priori, dans des recette ad hoc (pas d'intervention manuelle durant l'installation sur le device)
  - Assurer l'indépendance avec le PC hôte

# Reproductibilité et isolation

- Tous les composants utilisés (excepté des outils très basique tels shell, python, compilateur natif ...) viennent de la sandbox Yocto :
  - Outils natifs utilisés générés par Yocto
  - Headers et librairies croisées utilisés durant la compilation croisée
- L'utilisation d'une sandbox évite les erreurs dues à des headers incorrects
  - Les headers coté hôte n'ont pas forcément la bonne version pour les logiciels cross compilés pour la cible.

CAP'TRONIC

# Bon à savoir

- Outil historiquement en mode console :
  - Mais apparition de version graphique : Hob
  - Plugin Eclipse : ADT
- Prévoir beaucoup de disque et de temps CPU :
  - Génération de la toolchain + libc par Yocto (temps CPU)
  - Conservation des étapes intermédiaires – optionnel mais utile – très gourmand en disque
- Connaissance de Python : non obligatoire mais un + pour comprendre / développer des recettes
- Connaissance des standards tels que autotools, pkgconfig etc... conseillée :
  - Plus du fait des logiciels gérés que de Yocto lui même

CAP'TRONIC

# Conclusion

- Yocto fournit un framework complet pour développer des distributions embarquées riches et fiables :
  - Jeu de recettes large
  - Forte communauté d'utilisateurs
  - Investissement d'acteurs majeurs (fondeurs, vendeurs d'outils, sociétés de software ...)
- Le framework et la communauté prennent en compte la compléxité et la fragmentation de l'écosystème Linux
- Utilisé par des fondeurs majeurs (Intel, AMD, Freescale, T.I. ...), fondation de distribution Linux commerciales (Wind River Linux, Enea Linux, Mentor Graphics Embedded Linux ...)
- 2 release par an, sur une base prévisible (Printemps et Automne)

CAP'TRONIC

www.ciose.fr

# Conclusion

- Yocto est un outil de packaging, la fiabilité globale dépend également de la qualité des projets upstream
  - Mais des corrections locales
  - Un effort de sélection de projets « sérieux »
- Ne pas négliger qu'il n'y a pas d'outil miracle :
  - Temps de prise en main initial
  - Courbe d'apprentissage pour passer par les stades :
    - J'utilise
    - Je comprends
    - Je modifie / je crée
- La documentation s'est professionnalisée (sur le site Web Yocto et distribuée avec le code).
- Se faire accompagner par un spécialiste : réduction du Time To Market

CAP'TRONIC

www.ciose.fr

# Projets développés avec Yocto

- Projets développés par CIO, basés sur Yocto :
  - Device de gestion de l'énergie : base de données embarquée, FTP et Web server, communications Modbus
  - SCADA contrôlant la motorisation de bateaux : Qt, localisation et internationalisation (langues asiatiques)
  - Système In Flight Entertainment : codecs, protocoles vidéo et audio, DLNA, SOAP, librairies graphiques (moving map)
  - Appareil d'analyse médicale : Xorg sans Window manager, Java, serveur impression Cups, BDD Postgresql
  - Terminal de vote électronique : Qt4 embedded sur framebuffer, sécurisation des installations par cryptograpie asymétrique
  - Et bien d'autres en cours ou à venir ….

CAP'TRONIC

www.ciose.fr

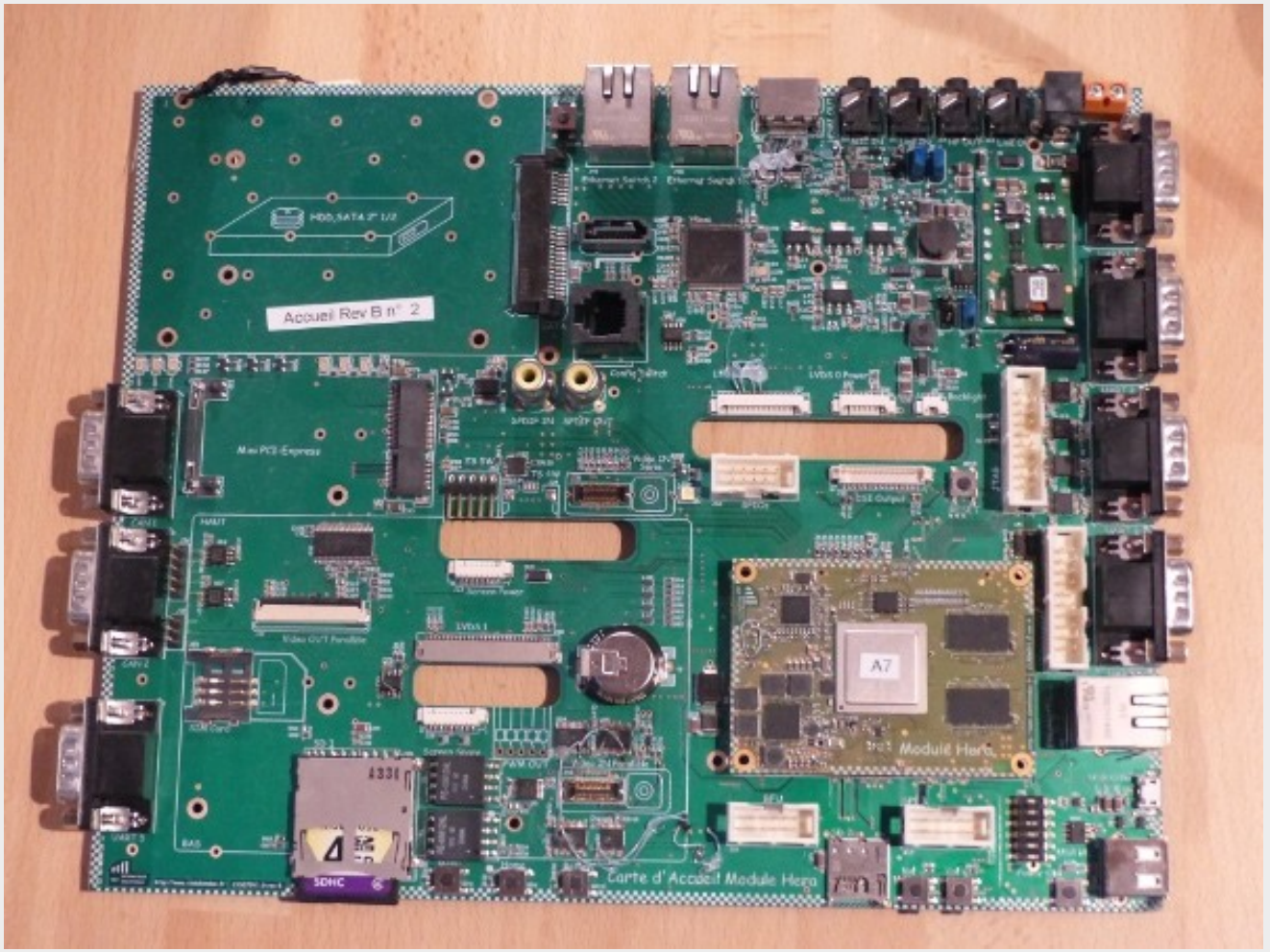
# Démonstration

- Démo effectuée sur COM Hera développé par CIO :
  - Processeur IMX 6 de Freescale
  - Distribution Yocto utilisée comme base des solutions In Flight Entertainment
  - Carte d'accueil = carte de développement
    - Remplacée par carte métier dédiée sur avion


CAP'TRONIC

www.ciose.fr

# Démonstration




CAP'TRONIC