UNIVERSITÉ
CÔTE D'AZUR

Synthèse du Cours

Présentation: Stéphane Lavirotte
Auteurs: ... et al*

(*) Cours réalisé grâce aux documents de :
Christophe Blaess, Stéphane Lavirotte

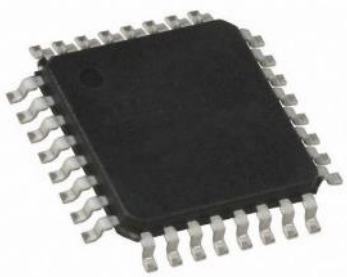
Mail: Stephane.Lavirotte@unice.fr
Web: http://stephane.lavirotte.com/
Université Nice Sophia Antipolis

UNIVERSITÉ CÔTE D'AZUR

Critères de Comparaison des Systèmes Embarqués

- ✓ Tous ces paramètres, en plus des plus classiques (fréquences, quantité de mémoire, ...), permettent de caractériser un système embarqué
  - Classe de processeur:
    - Microprocesseur / Microcontrôleur
  - Gestion de la mémoire:
    - MMU / sans MMU
  - Système d'exploitation:
    - OS / sans OS
  - Type de système d'exploitation
    - Normal / Temps Réel
- ✓ Quand choisir quoi ?

UNIVERSITÉ
CÔTE D'AZUR



Unité de Calcul

Microcontrôleur - Microprocesseur

UNIVERSITÉ CÔTE D'AZUR

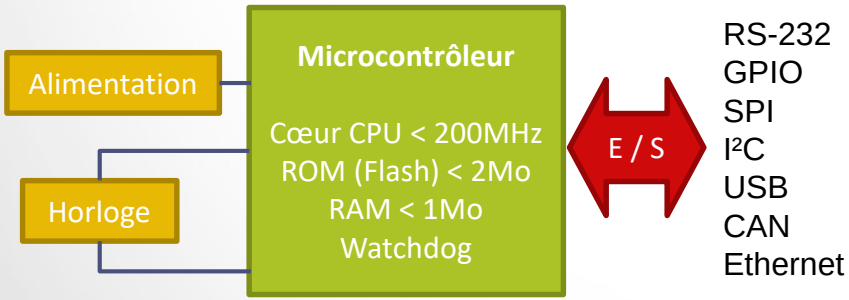
µContrôleur - µProcesseur

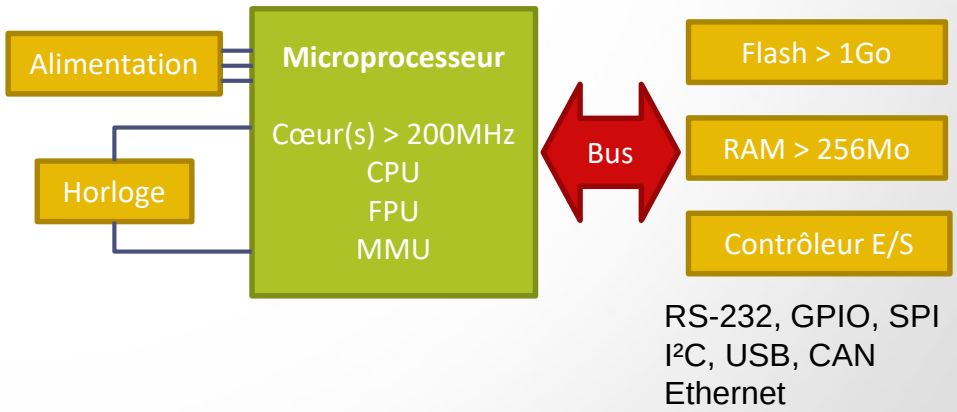
Microcontrôleur

- ✓ Electronique simple
- ✓ Déterminisme
- ✓ Fiabilité fonctionnement
- ✓ Généralement sans OS

Microprocesseur

- ✓ Electronique complexe
- ✓ Entrées-sorties par des contrôleurs externes
- ✓ Utilisation d'un OS

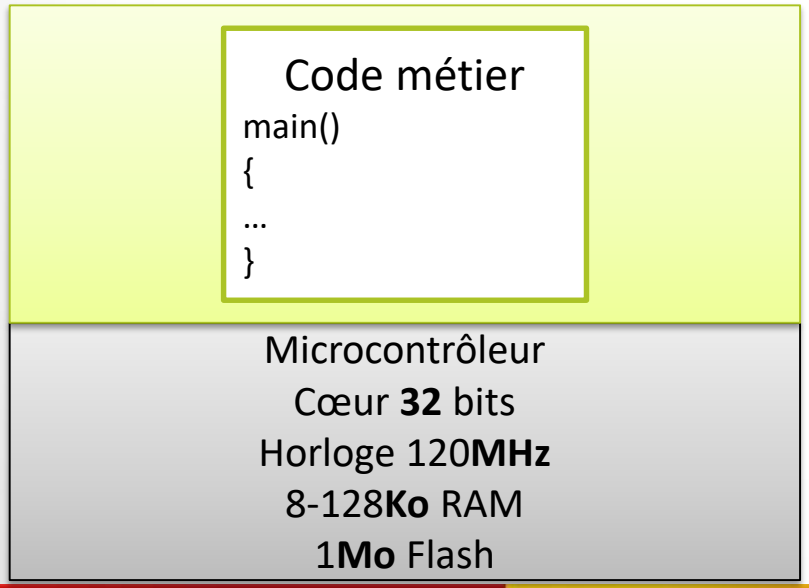




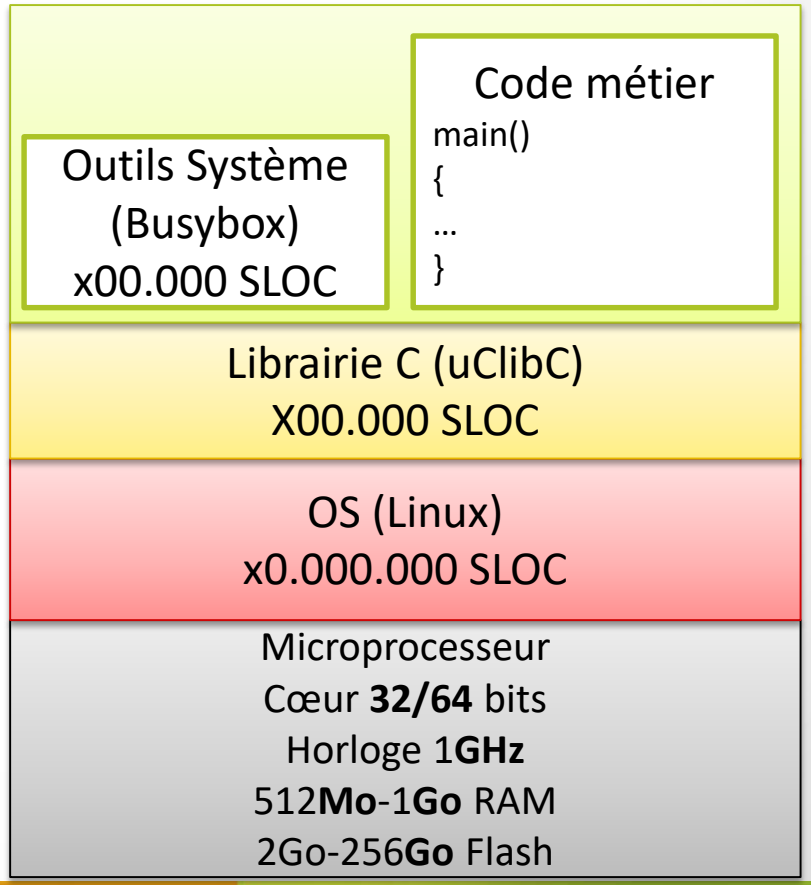
UNIVERSITÉ CÔTE D'AZUR

Ressources nécessaires

Système à Microcontrôleur



Système à Microprocesseur

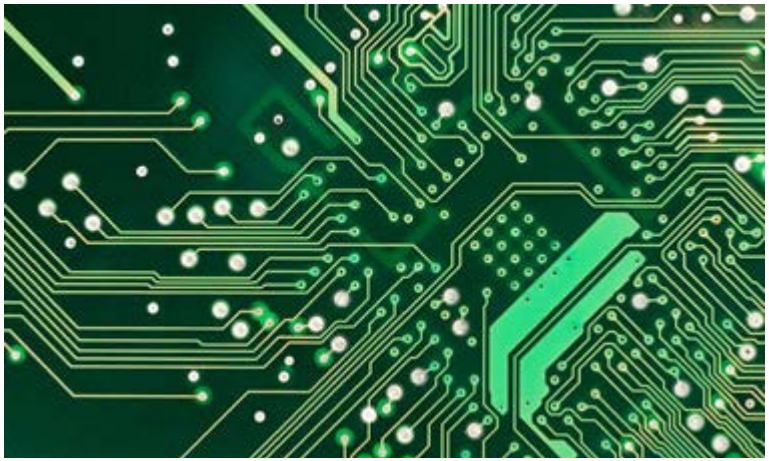


UNIVERSITÉ CÔTE D'AZUR

Choix d'une architecture matérielle

- ✓ Microcontrôleur
  - Prix
    - Conception et réalisation PCB, coût unitaire
  - Simplicité
    - Fiabilité, code certifiable
  - Prédictibilité
    - Temps d'exécution, déterminisme
- ✓ Microprocesseur
  - Puissance
    - Calcul, mémoire, optimisation
  - Evolutivité
    - Isolation du code métier / au matériel, portabilité
  - Richesse applicative
    - Pile de protocoles, services

UNIVERSITÉ
CÔTE D'AZUR

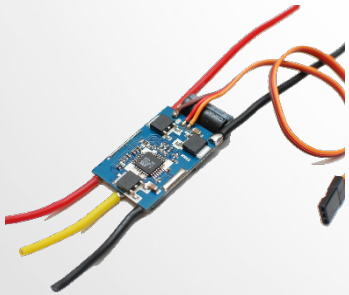


Architecture Matérielle

UNIVERSITÉ CÔTE D'AZUR

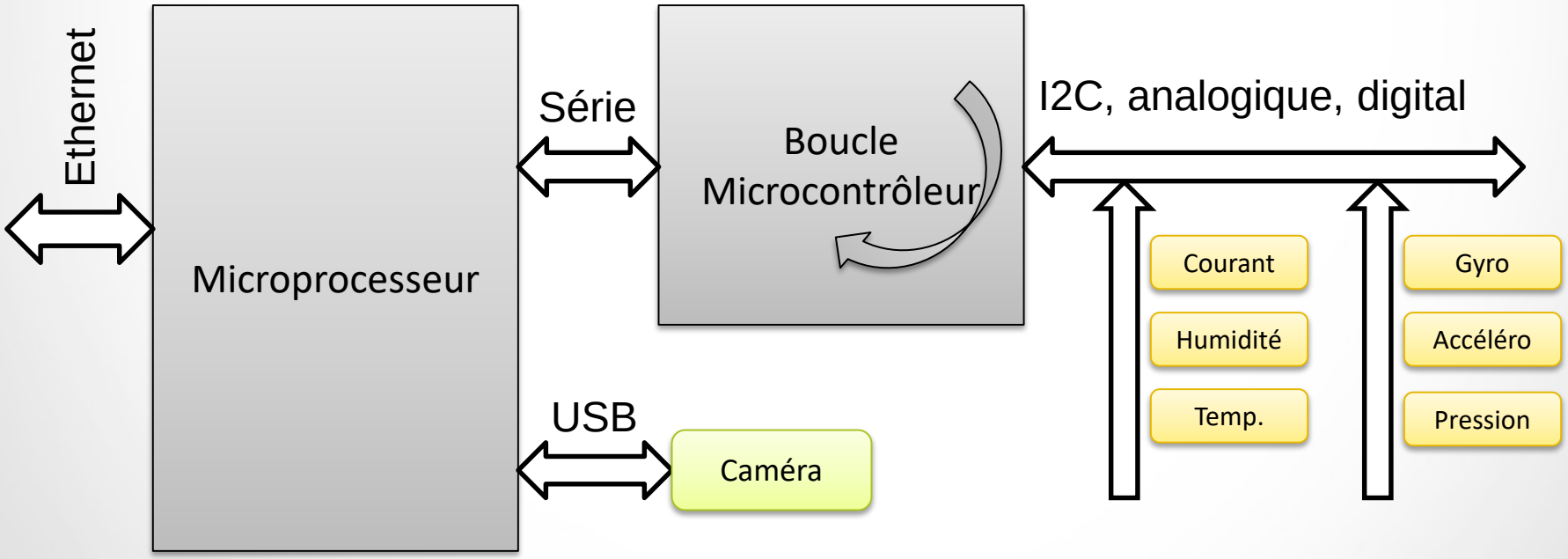
Exemples de projets: qui utilise quoi ?

1. Contrôleur de la vitesse d'un moteur avec accélération, décélération, ...
2. Montre connectée permettant de détecter l'activité de la personne qui l'a porte (capteurs: accéléromètre, gyroscope, tensiomètre) et un afficheur graphique
3. Caméra IP avec un serveur Web et la possibilité de faire des traitement d'image et de retransmettre le flux vidéo
4. Sous-marin intégrant des capteurs de distance, température, barométrique, ..., un serveur Web et réalisant la retransmission du flux vidéo de la caméra embarquée vers un client



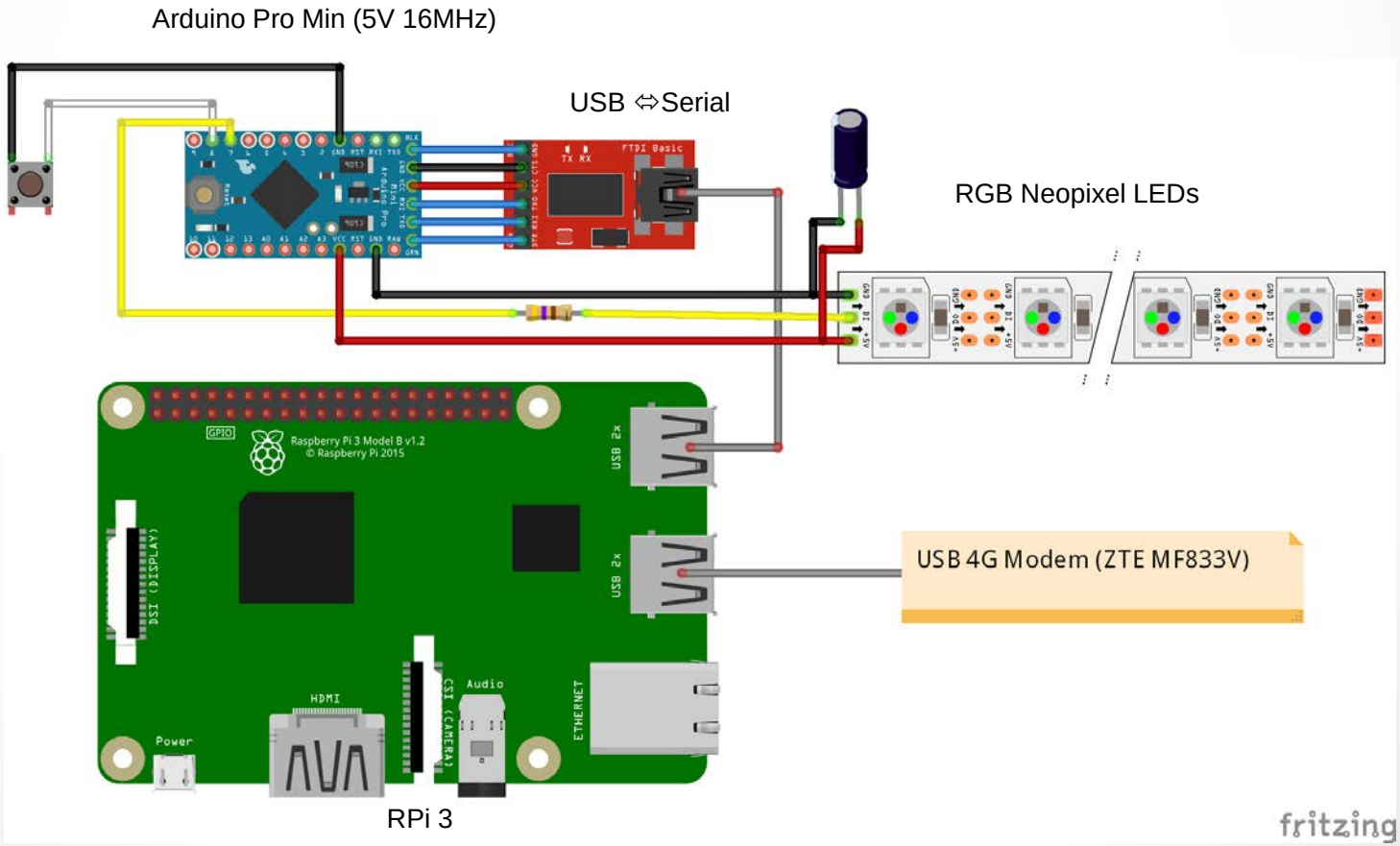
On a besoin du meilleur des 2 mondes

- ✓ **Architecture mixte**
  - **Microprocesseur - Microcontrôleur**



UNIVERSITÉ CÔTE D'AZUR

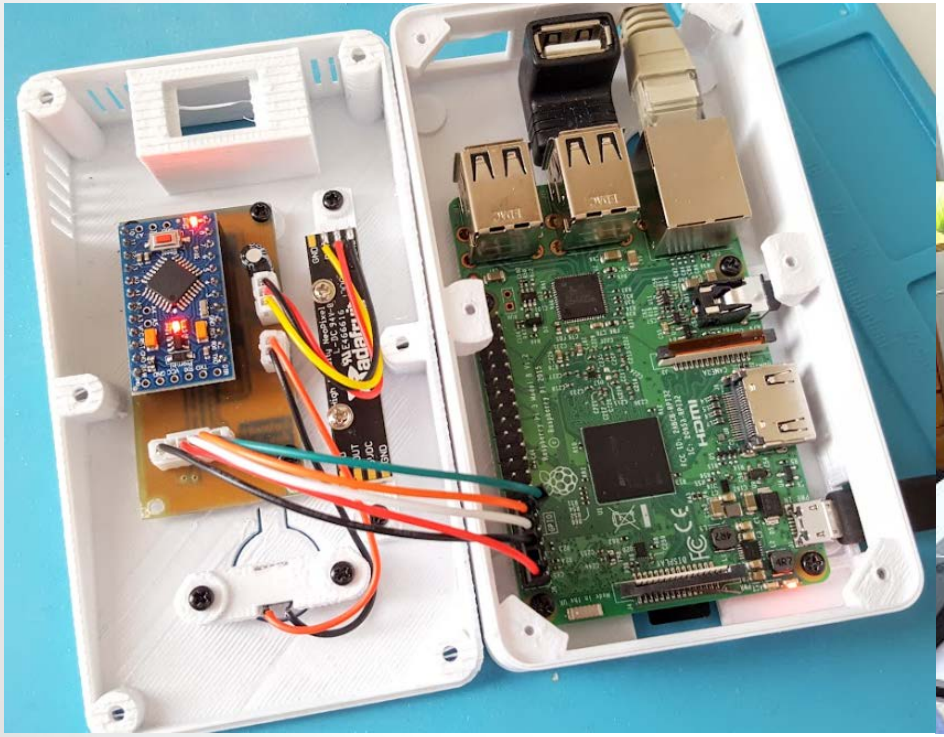
Exemple concret d'architecture mixte...



© Franck Fleurey – Tell U

UNIVERSITÉ CÔTE D'AZUR

… utilisé dans les prototypes suivants



tell.u®

http://www.tellucloud.com/

UNIVERSITÉ CÔTE D'AZUR

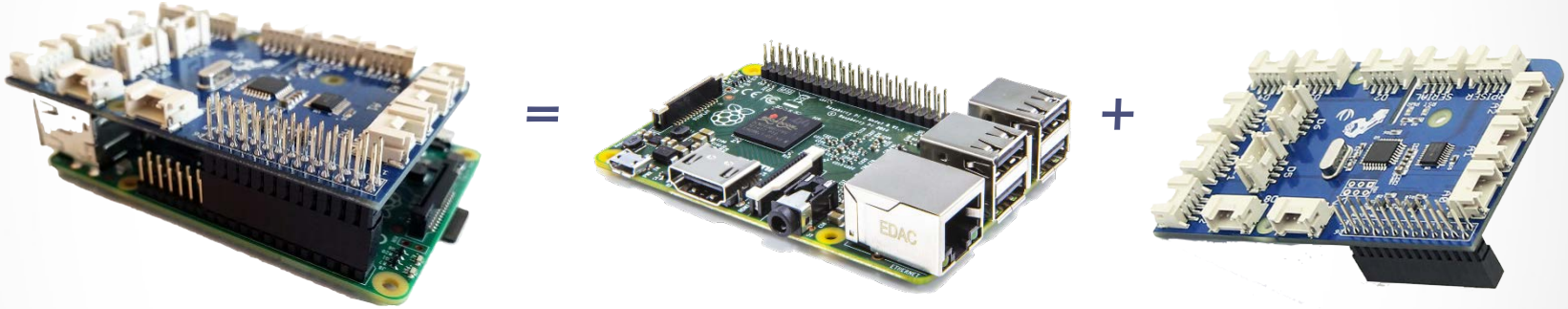
Architecture mixte

- ✓ Cette solution mixte présente des avantages...
  - Le microcontrôleur ajoute des GPIO, entrées analogiques, UART, CAN, ...
  - Code du microcontrôleur verrouillable pour être illisible (fusible)
  - Possibilité de verrouiller l'applicatif Linux pour qu'il ne tourne qu'avec une architecture matérielle donnée (protège contre la copie sans violer la GPL)
  - Possibilité de faire surveiller le CPU par le microcontrôleur
- ✓ ... et quelques inconvénients
  - Complexité d'une double unité de calcul dans l'architecture
  - Conception et réalisation d'un protocole de communication entre microprocesseur et microcontrôleur
  - Surcoût (oui, mais assez faible aux vues des avantages)

UNIVERSITÉ
CÔTE D'AZUR

Exemples de Plateformes

- ✓ SoC + Microcontrôleur = Raspberry Pi + GrovePi



- ✓ SoC + Microcontrôleur = Intel Edison Arduino

- ✓ Une approche qui s'étend dans l'IoT

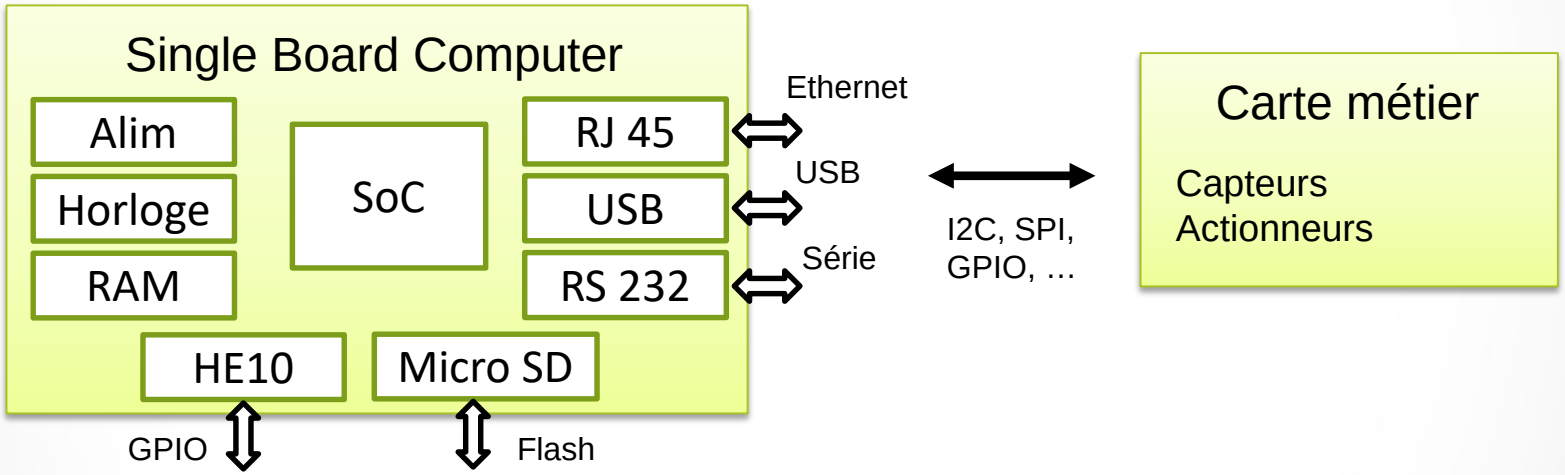
UNIVERSITÉ CÔTE D'AZUR

Choix Matériels Fonction du Contexte

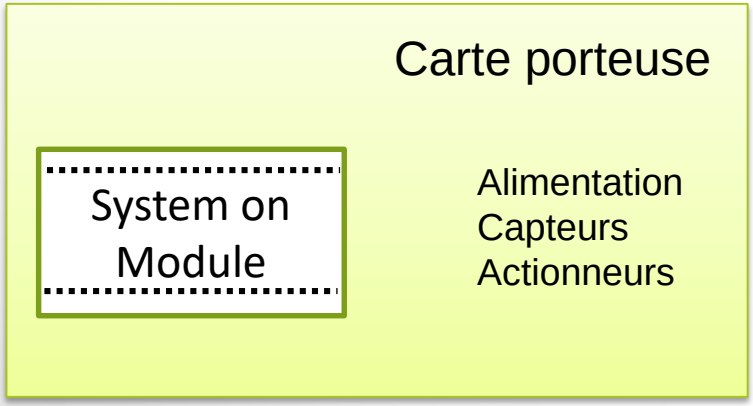
- ✓ Prototype – Projet Personnel
  - *Single Board Computer* (SBC)
    - Intégrant SoC, mémoire, E/S (GPIO, I2C, SPI, ...), microprocesseur (Raspberry) ou microcontrôleur (Arduino)
- ✓ Petite série – Startup
  - *Computer on module* (CoM) – System on Module (SoM)
    - Petite carte de dimension réduite contenant l'équivalent d'un ordinateur mono-carte (S.O.C, mémoire...) sans connecteurs
    - Carte porteuse intégrant alimentation, capteurs et actionneurs
- ✓ Grande série – Production industrielle
  - Intégration d'un System on Chip (SoC)
    - Carte spécifique intégrant tout
    - Nécessite un coût important pour le design, le test, et la validation (rarement intéressent en dessous de dizaine de milliers d'unités)

Plateformes Fonction du Contexte

✓ Prototypage



✓ Petites séries



Grandes séries

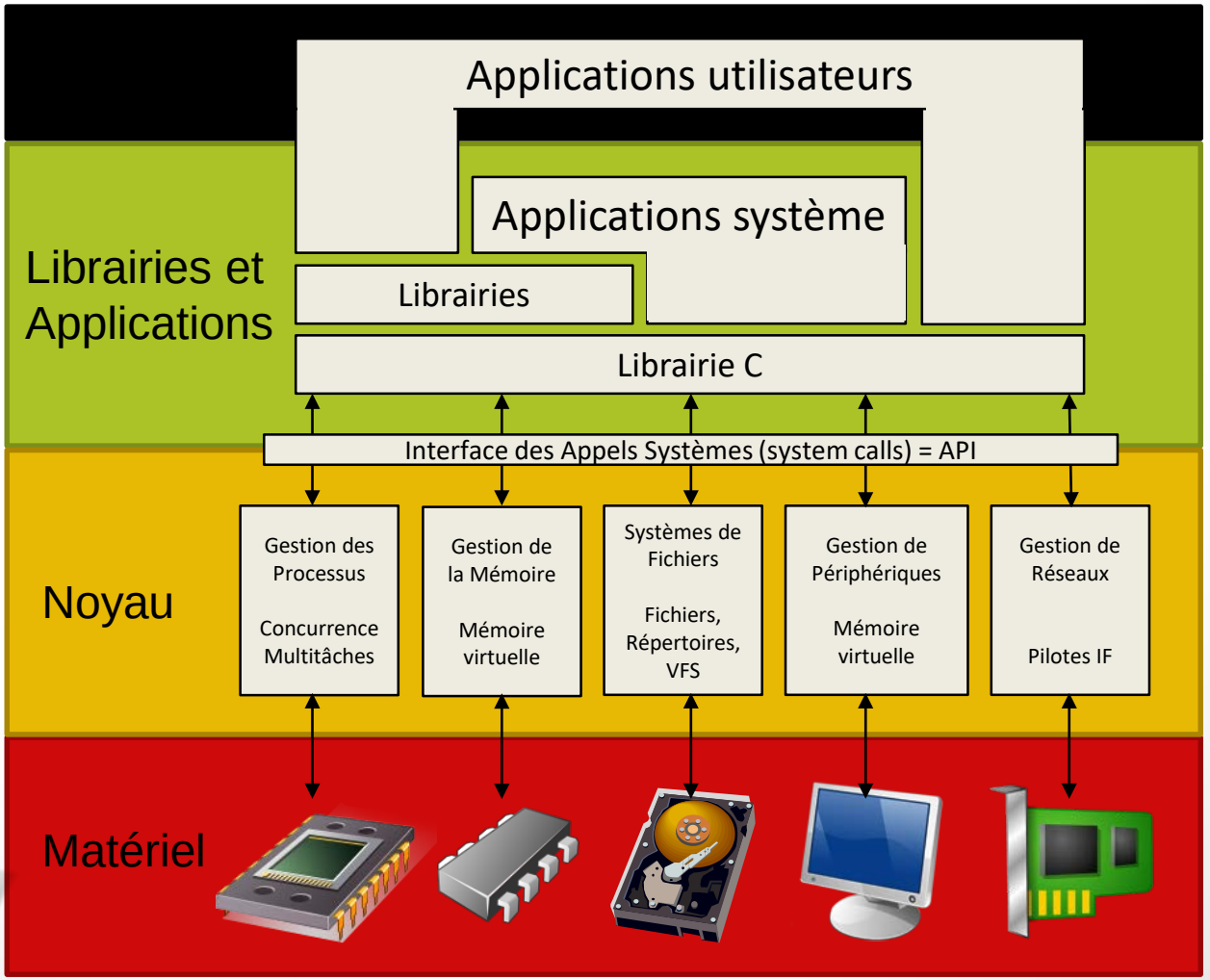
UNIVERSITÉ
CÔTE D'AZUR

Avec ou sans OS, normal ou RT

Systèmes d'Exploitation et spécificités de l'embarqué

UNIVERSITÉ CÔTE D'AZUR

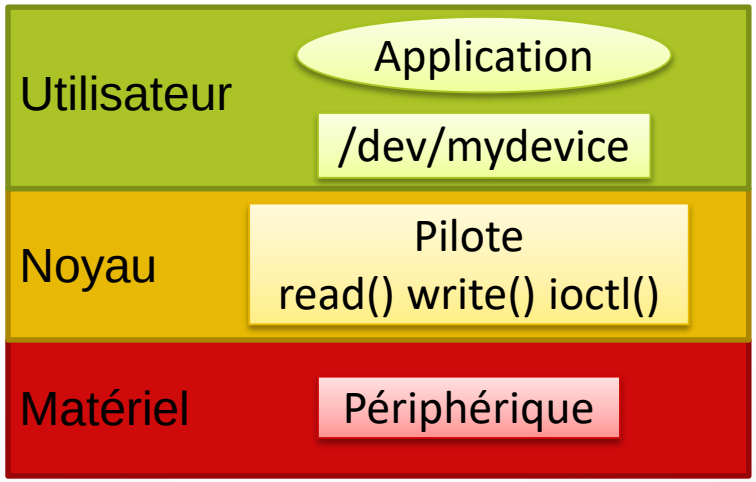
Architecture générale d'un Système d'Exploitation



UNIVERSITÉ CÔTE D'AZUR

Abstraction des Périphériques

- ✓ Le support pour les périphériques est assuré par les pilotes (drivers)
  - Dans le noyau ou externe au noyau (en module ou espace utilisateur)
- ✓ Les applications communiquent avec les périphériques en réalisant des opérations (lecture, écriture sur des pseudos fichiers) (`/dev/...`)

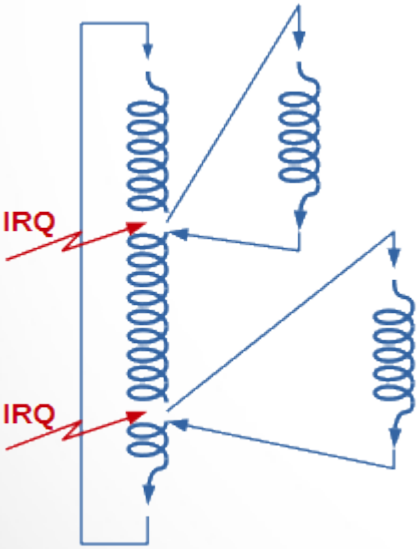


UNIVERSITÉ CÔTE D'AZUR

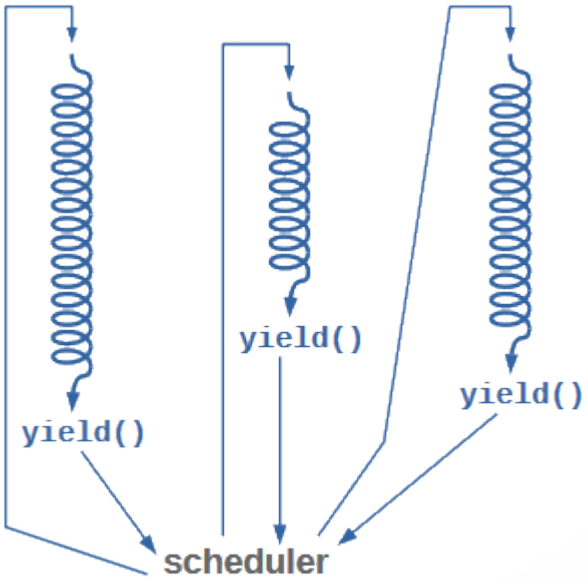
Apport d 'un Système d'Exploitation

- ✓ Exécution des tâches
  - L'ordonnanceur (*scheduler*) est une fonctionnalité essentielle des OS pour exécuter les tâches sur un même processeur

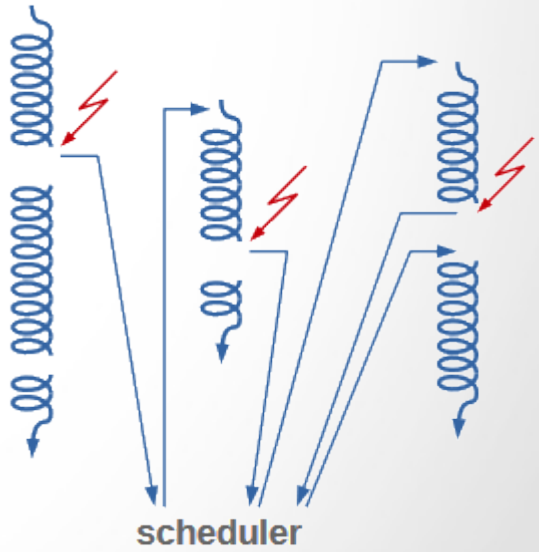
Mono tâche



Mono tâche + Interruptions



Multitâches coopératif



Multitâches préemptif

UNIVERSITÉ CÔTE D'AZUR

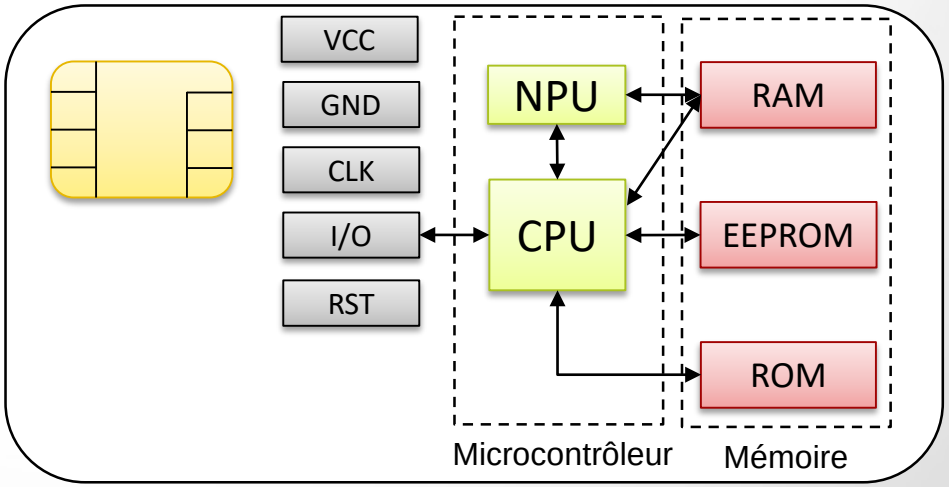
Systèmes d'Exploitation pour l'Embarqué

✓ Pour réseaux de capteurs

OS	Architecture	Modèle Prog.	Ordonancement	Communiation	Temps Réel
TinyOS	Monolithique	Evénementiel	FIFO	Active Msg	Non
Contiki	Modulaire	Protothreads + Evénementiel	Par priorité WRT	uIP et Rime	Non
LiteOS	Modulaire	Threads + Evénementiel	Priorité + modèle Round Robin	Par Fichiers	Non

✓ Cartes à Puce

- JavaCard
- MulOS

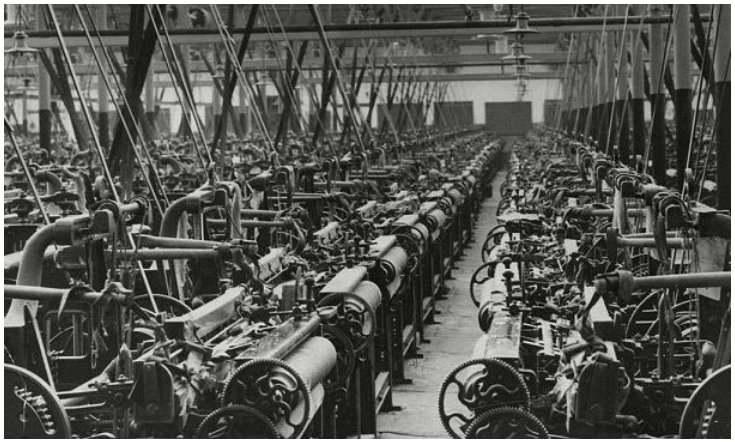


UNIVERSITÉ CÔTE D'AZUR

Temps Réel libre pour les Systèmes Embarqués

- ✓ Temps réel souple (soft realtime) : Linux Vanilla
  - Contraintes temporelles en **millisecondes**
  - Comportement moyen, par de garantie (best effort)
- ✓ Temps réel souple renforcé: Linux avec PREEMPT_RT
  - Contraintes temporelles en centaines de **microsecondes**
  - Comportement prévu géré dans les pire circonstances (worst cases)
- ✓ Temps réel strict (hard realtime)
  - Non certifiable: Linux avec Xenomai
    - Contraintes temporelles en **dizaines de microsecondes**
    - Comportement dans le pire cas vérifié
  - Certifiable: RTEMS, FreeRTOS
    - Contraintes temporelles en **microsecondes**
    - Code vérifiable (code minimal)
- ✓ Temps réel absolu: FPGA, logique électronique
  - Contraintes en dizaines de **nanosecondes**

UNIVERSITÉ
CÔTE D'AZUR

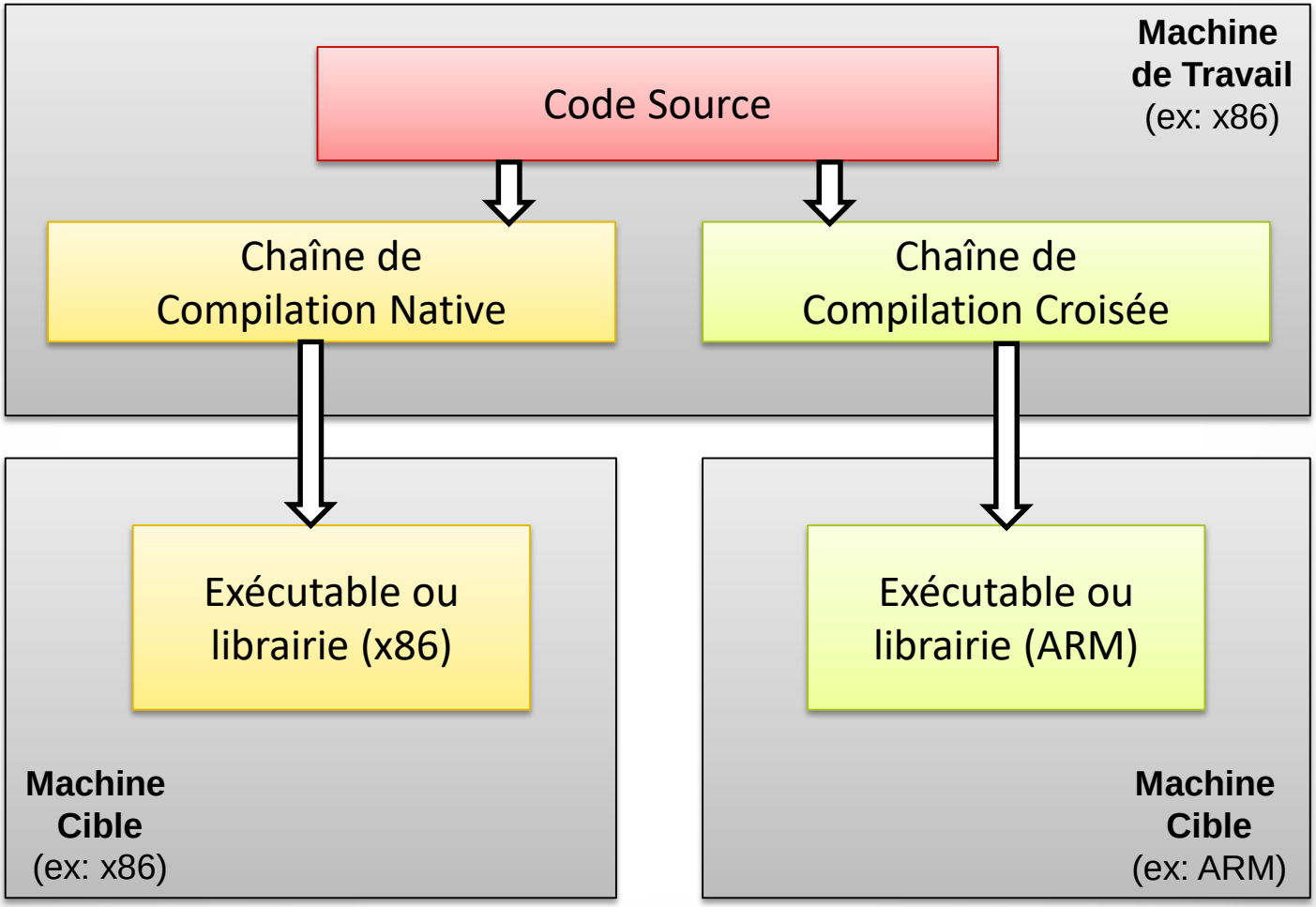


Logiciels pour l'Embarqué

Vers une industrialisation du processus

UNIVERSITÉ CÔTE D'AZUR

Compilation Croisée: Générer du Code pour une autre Architecture



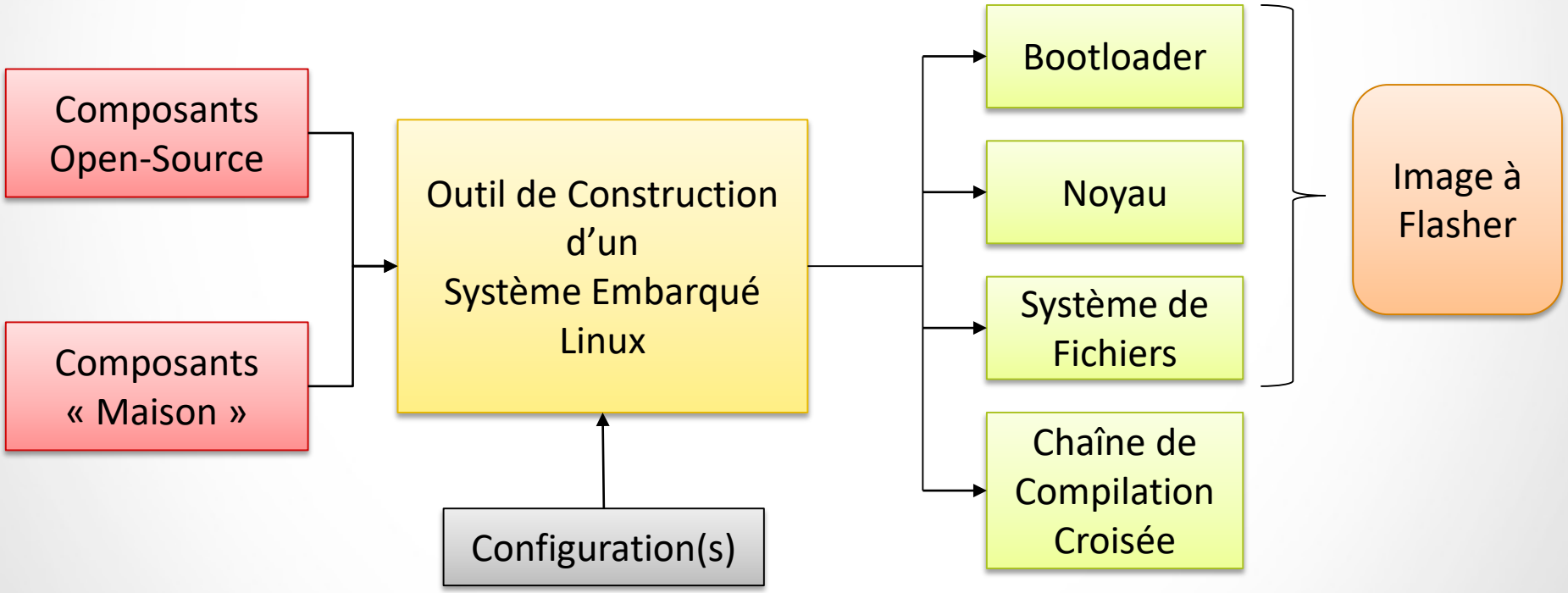
UNIVERSITÉ CÔTE D'AZUR

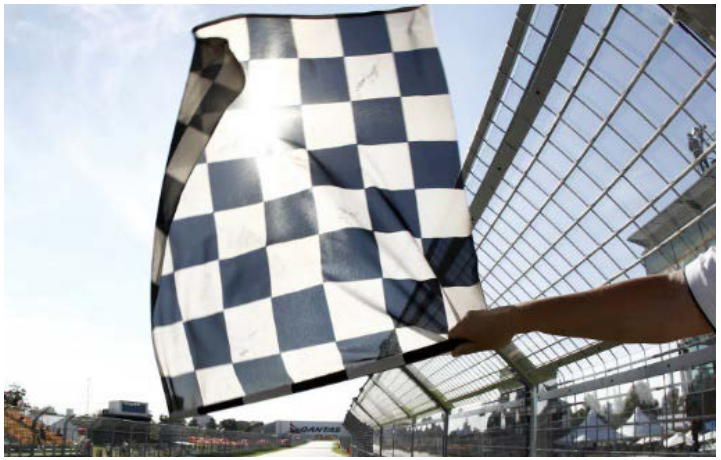
Choix Logiciels Fonction du Contexte

- ✓ Prototype – Projet Personnel
  - Utilisation d'une distribution Linux précompilée
  - Prototypage rapide par scripts
  - Inscription du code métier dans les scripts de démarrage
  - Démonstration, Proof of Concept
- ✓ Industrialisation (petites ou grandes séries)
  - Industrialisation du processus logiciel crucial
    - Outils logiciels classiques (intégration continue, gestion de versions, ...)
  - Utilisation de boîtes à outils pour l'embarqué (Bysybox)
  - Utilisation de systèmes de construction (Buildroot, Yocto, ...)
  - Configuration du noyau optimisée (temps de boot, occupation mémoire, choix des drivers)
  - Mécanisme de déploiement et de mise à jour du système

UNIVERSITÉ CÔTE D'AZUR

Principe de Production d'un Système Embarqué avec OS





Conclusion

Il faut bien finir un jour !

UNIVERSITÉ
CÔTE D'AZUR

Conclusion

- ✓ Plusieurs facteurs sont à prendre en compte lors de la conception d'un système embarqué
  - Puissance de calcul nécessaire, quantité de mémoire
  - Capteurs et actionneurs nécessaires
  - Types d'entrées-sorties
  - Contraintes physiques (dimensions, poids, autonomie, ...)
  - Nécessité d'un OS et choix éventuel (RT ou pas)
  - Méthodes de développement logiciel
  - Déploiement et mise à jour du code
- ✓ Les choix peuvent évoluer au cours de la mise au point du projet
  - La plateforme utilisée pour le prototype ne sera pas la même que la première série ou la production en nombre
- ✓ Mais attention à faire des choix raisonnés et cohérents!