Capitole du Libre 2014

Du microcontrôleur au système Linux embarqué

Choix d'architectures matérielles et logicielles

Christophe Blaess
http://christophe.blaess.fr
http://www.logilin.fr

Ingénierie et formations Linux industriel

Blog : christophe.blaess.fr

Microprocesseur vs microcontrôleur

Introduction - terminologie
Différences matérielles
Spécificités logicielles

Choix d'architecture

Prototypage – Projet personnel
Petite série – Startup
Grande série – production industrielle

Étude de cas

Problématique
Système à microcontrôleur
Microprocesseur sous Linux
Architecture hybride
Projet LxMCU

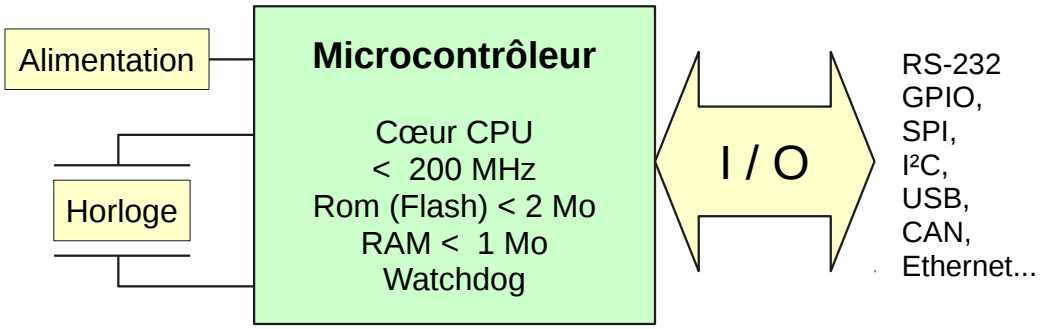
Conclusion

Questions ?

Microprocesseur *vs* microcontrôleur

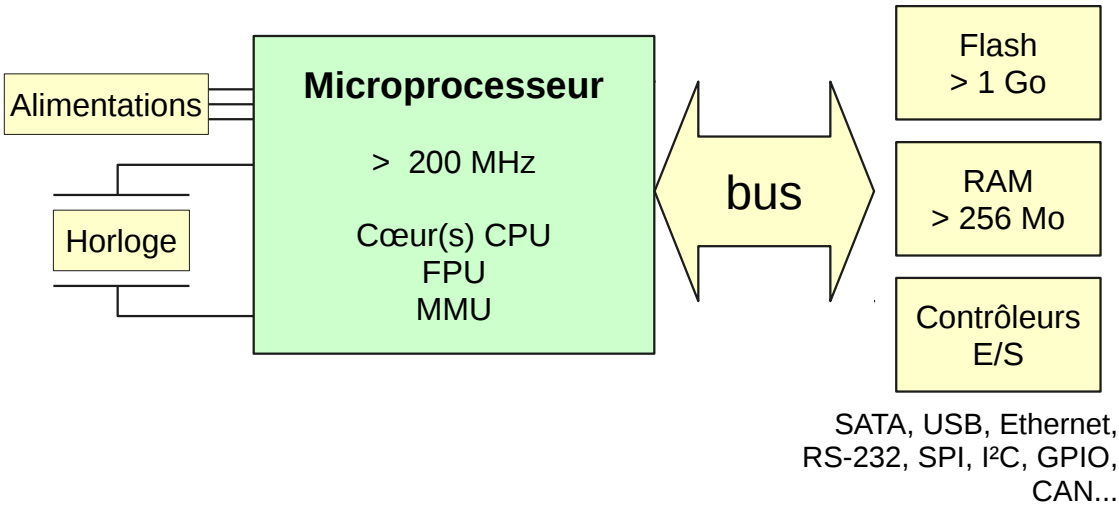
Introduction - terminologie

Microcontrôleur (µC)



- Mise en œuvre électronique simple.
- Déterminisme et fiabilité de fonctionnement.
- Généralement pas de système d'exploitation (ou minimal).

Microprocesseur (µP)

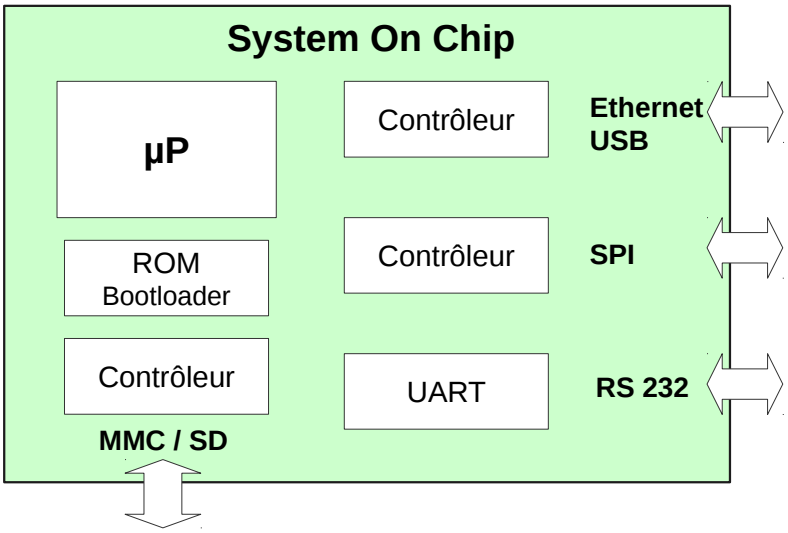


Entrées-sorties réalisées par des contrôleurs externes au processeur

Mise en œuvre électronique beaucoup plus complexe

Optimisé pour l'utilisation d'un système d'exploitation

System on Chip (S.O.C.)



Contrôleurs d'entrées-sorties déjà incorporés

Intégration électronique encore assez complexe

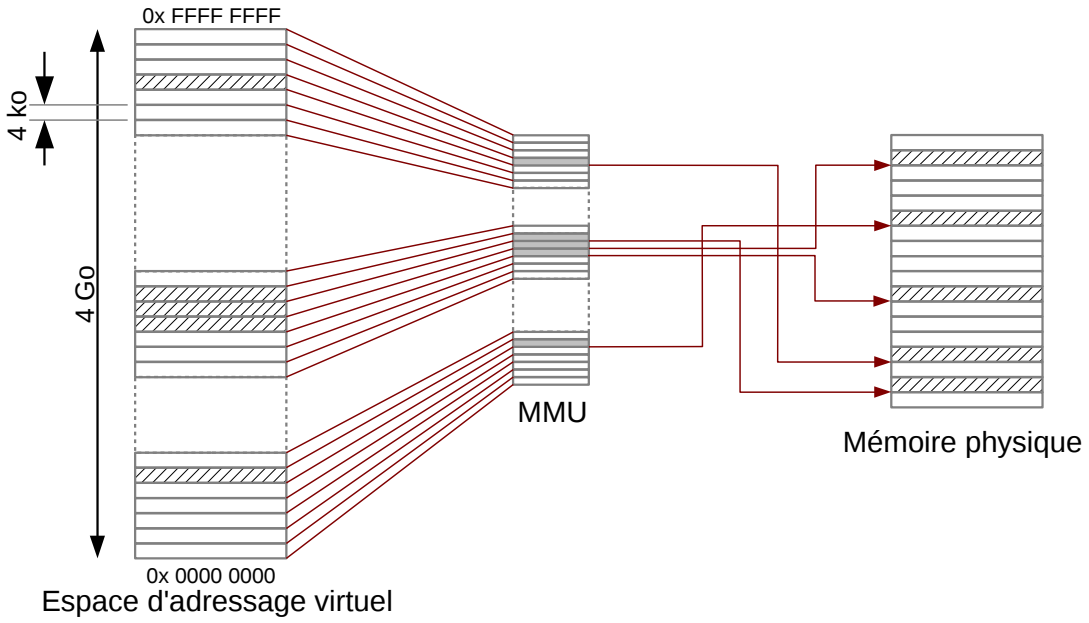
Souvent peu d'entrées-sorties industrielles (CAN) ou analogiques (ADC/DAC, PWM)

Différences matérielles

Comparatif

	Microcontrôleur	***System-on-chip***
Coût moyen	< 10 €	> 20 €
PCB support	Simple – 2 couches	Complexe – 6 couches
Alimentations	Simple – 3.3 V	Multiples 3.3V / 5V / 12 V
Volume de code métier	~ 100 ko	> 10 Mo
Environnement de développement	Propriétaire > 3000 € Libre (Gnu) pas toujours disponible.	Libre (Gnu) et gratuit si développement sous et pour Linux.
Mise au point du code	Complexe. Débogueur spécifique.	Plus simple grâce à l'O.S.
Déploiement, mise à jour	Complexe.	Simple. Plusieurs possibilités.
Protection du code métier	Facile (fusibles).	Difficile.

Mémoire virtuelle et MMU



Un processus voit un espace de mémoire virtuelle, au sein du quel il peut accéder à n'importe quelle adresse de `0x000000` à `0xFFFFFFFF` (sur processeur 32 bits).

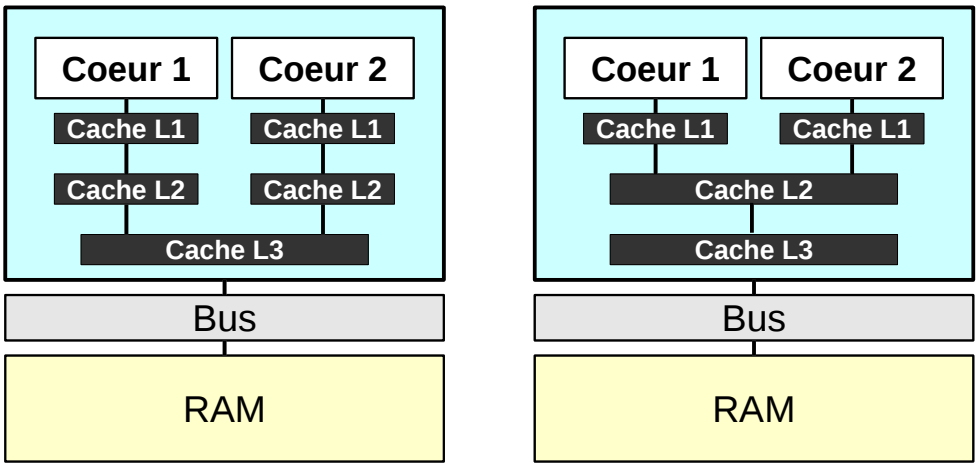
Cet espace est découpé en pages, et la MMU – Memory Managment Unit (un composant intégré dans le processeur) – associe une page de mémoire virtuelle avec une page de mémoire physique en effectuant la modification d'adresse lors de l'accès à la mémoire.

Certaines pages de mémoire virtuelle n'ont pas de correspondance en mémoire physique : une tentative d'accès déclenche une interruption « faute de page ».

Chaque processus dispose d'une configuration personnelle de la MMU. Cette dernière est programmée à chaque commutation entre deux processus.

Un processus ne voit que les pages de mémoire physique qui lui ont été attribué par le noyau ; les pages des autres processus ne sont projetées à aucun emplacement de sa mémoire virtuelle.

Systèmes multicœurs

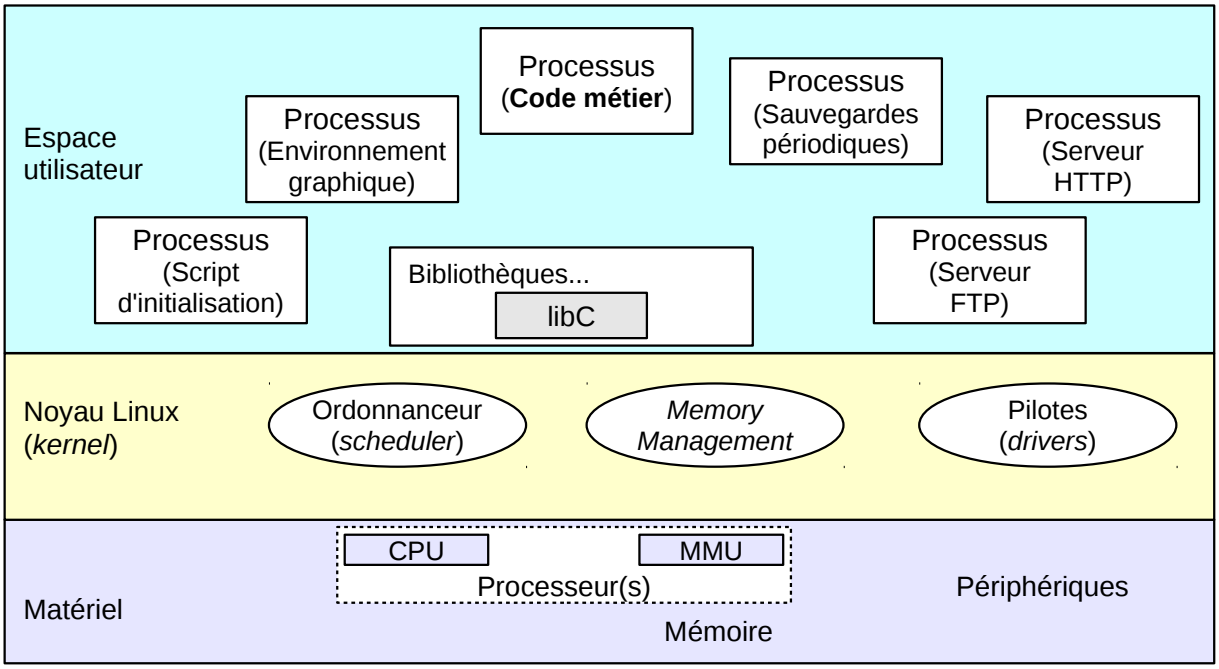


L'unité de traitement des données est multipliée (par deux ou quatre en général).

Le cache de premier niveau est spécifique à chaque cœur. Le cache de second niveau peut être partagé ou non. Lorsque le cache de niveau 2 est partagé l'accès est plus rapide, sinon il faut des messages de synchronisation lors d'accès à des zones identiques depuis les deux cœurs.

Spécificités logicielles

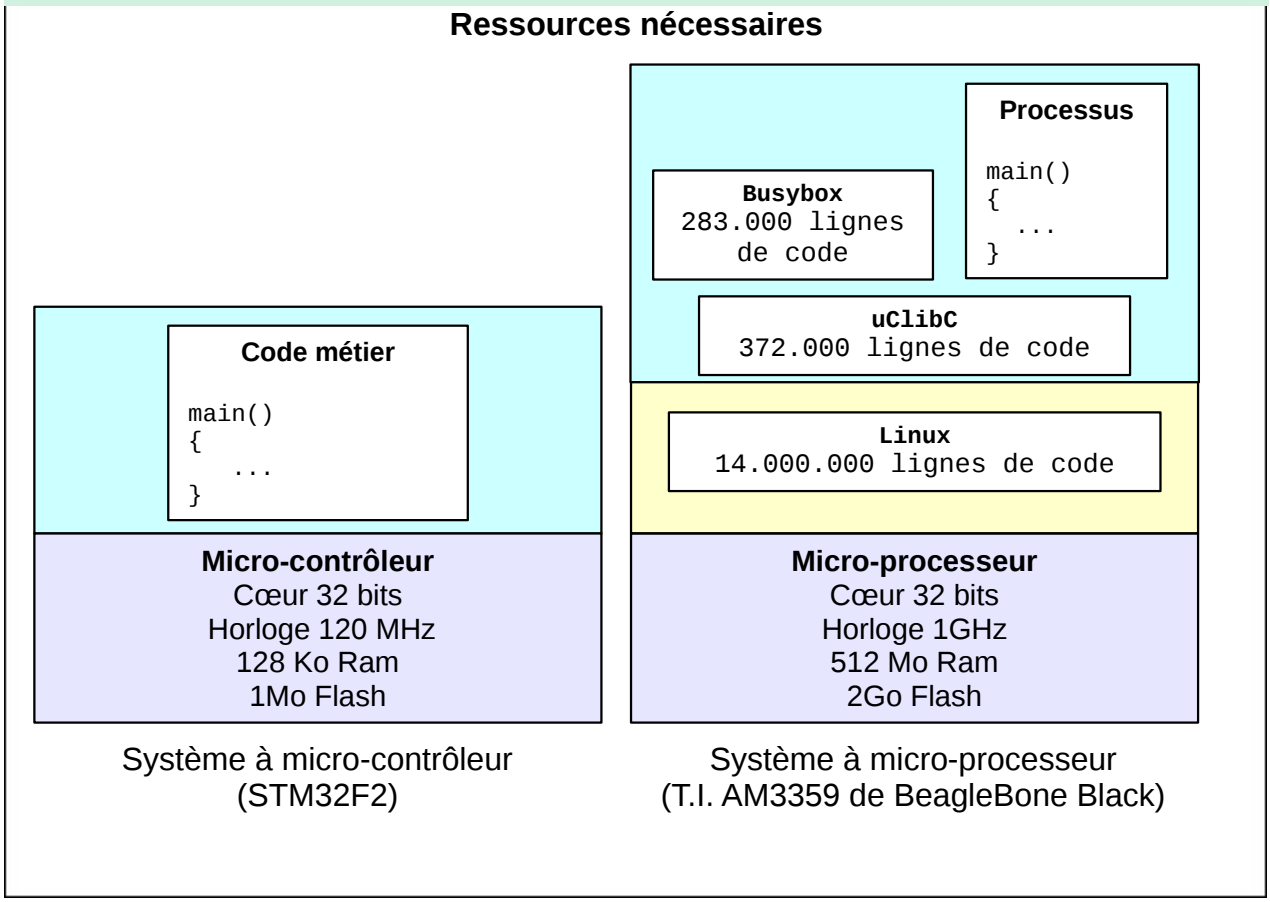
Composants d'un système Linux embarqué



Le rôle du noyau Linux : mettre les ressources offertes par le matériel à disposition des applications de l'espace utilisateur.

Lorsqu'il exécute le code du noyau, le processeur est en mode superviseur (privilégié). Pour exécuter du code en espace utilisateur, il passe en mode protégé (non-privilégié).

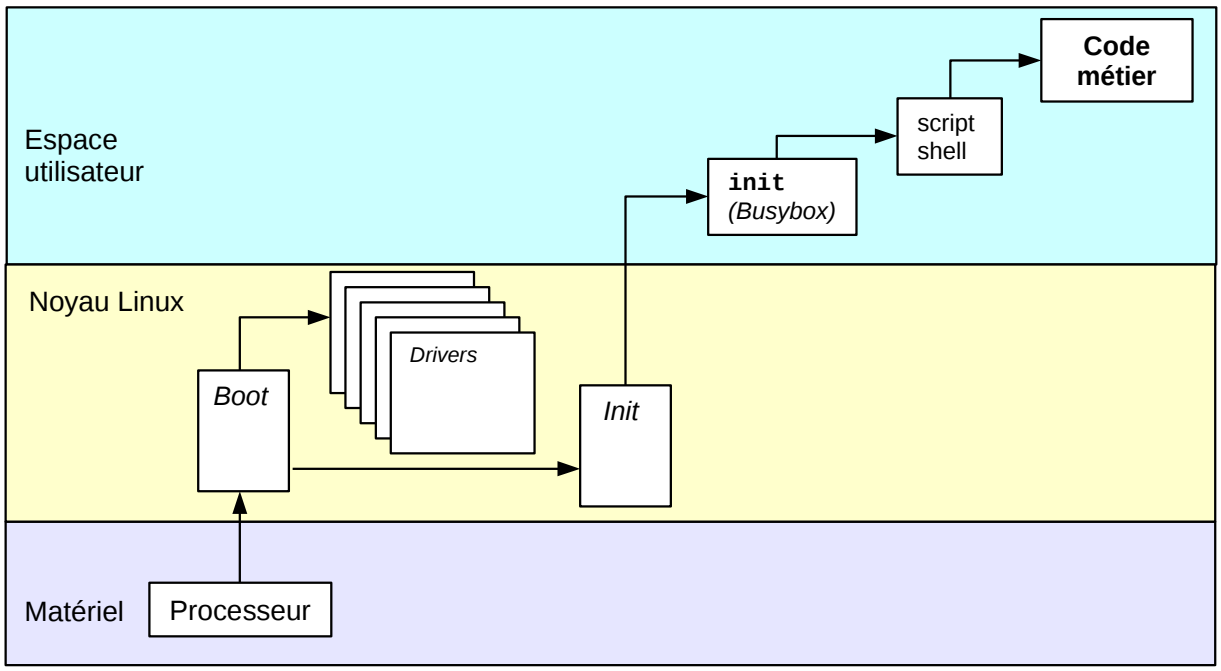
Les tâches (*threads*) de l'espace utilisateur s'exécutent dans des processus (espaces de mémoire disjoints).



Sur un système à micro-contrôleur, le code métier est le seul maître à bord, il accède à volonté aux périphériques, à la mémoire, etc.

Au contraire, dans un système à micro-processeur, le code métier n'est qu'une petite partie de l'ensemble du logiciel. Il est soumis à l'ordonnancement et au sous-système de gestion mémoire du noyau. Il s'appuie sur des bibliothèques et des utilitaires externes.

Temps de boot



Suivant le type de processeur et la complexité du matériel, le temps de *boot* du noyau dure de deux à cinq secondes environ.

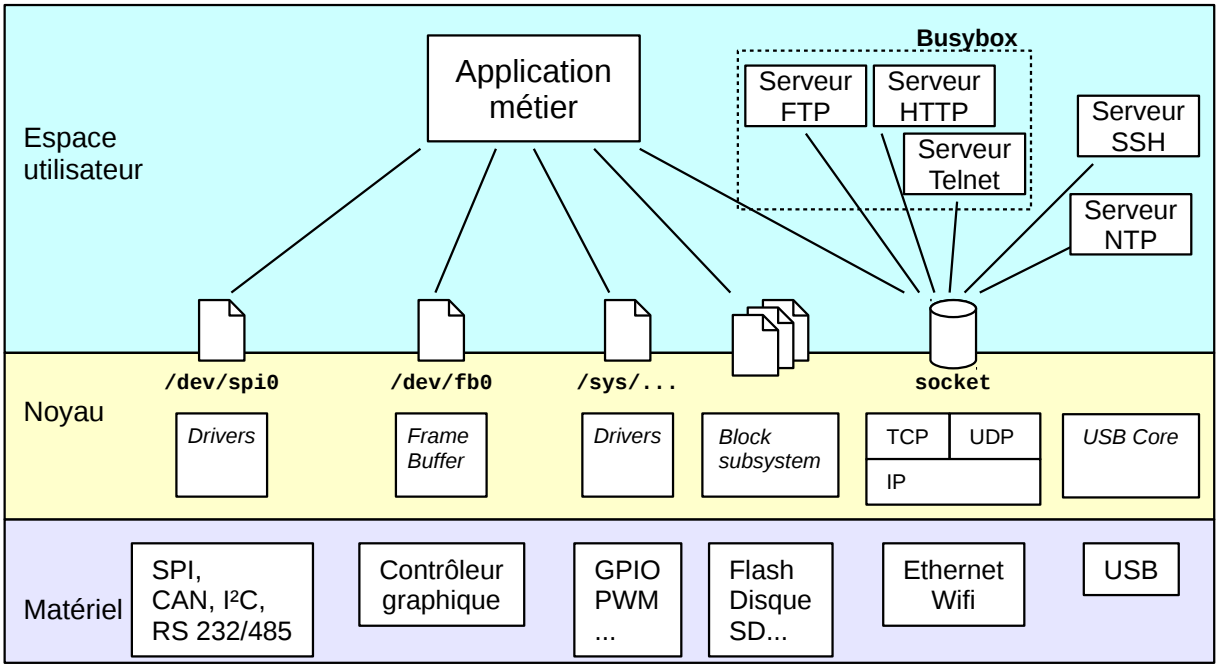
Le démarrage du processus `init`, les tâches administratives (montage systèmes de fichiers, configuration paramètres de `/proc`, etc.) prennent une à deux secondes supplémentaires.

Le lancement de tous les services (réseau, authentification, environnement graphique, etc.) peut demander une dizaine de secondes.

Pour en savoir plus

« *Optimisation du temps de boot d'un système Linux embarqué* » Christophe Blaess – Open Silicium 9, voir http://www.blaess.fr/christophe/articles/ .

Services offerts par le noyau Linux



Le noyau Linux permet directement d'accéder à plusieurs centaines de périphériques et protocoles de communications, de réaliser des opérations d'entrées-sorties aisément, d'afficher une interface graphique, de lire des dizaines de formats de systèmes de fichiers, de dialoguer avec de nombreux protocoles réseau, etc.

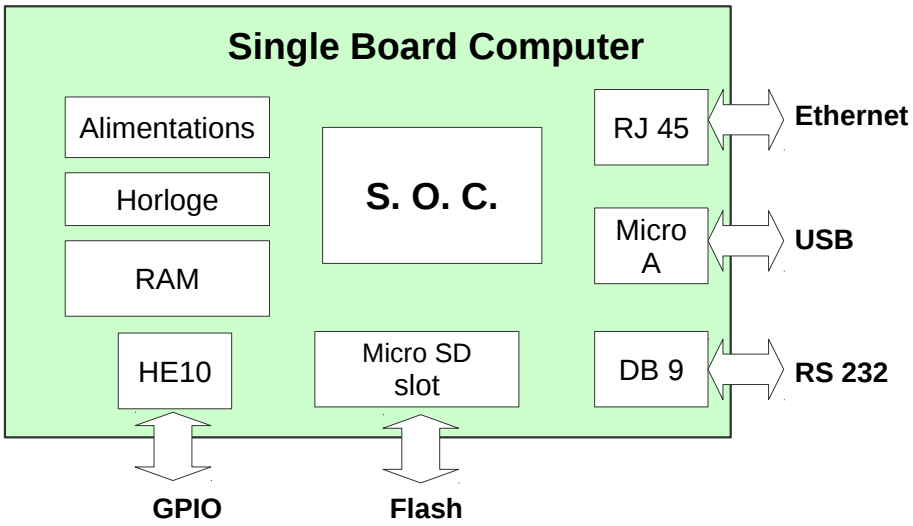
Si un élément n'est pas présent dans une configuration embarquée, il suffit généralement de recompiler le noyau (voire de ne compiler qu'un module) pour l'intégrer.

Choix d'architecture

Prototypage – Projet personnel

SBC (Single-Board-Computer)

Ordinateur mono-carte intégrant *system-on-chip*, mémoire, connecteurs d'E/S, etc.



Exemples : BeagleBone Black, Cubieboard, Raspberry Pi, OLinuXino...

Certains SBC (sans Linux) reposent sur des microcontrôleurs : Arduino, Launchpad, etc.

Environnement de développement

Utilisation de distributions Linux pré-compilées :

- Raspbian pour Raspberry PI
- Arch Linux,
- etc.

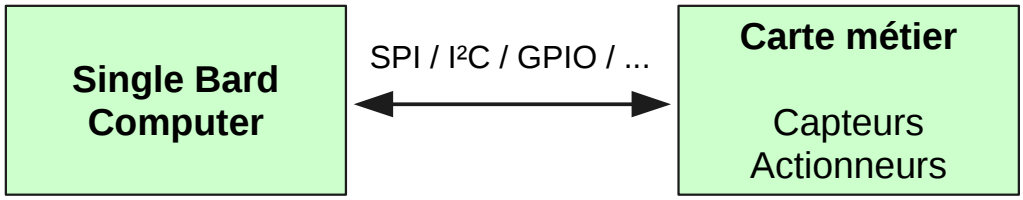
Développement du code métier directement sur la cible.

Prototypage rapide par scripts (Python par exemple).

Inscription du code métier dans les scripts de démarrage de l'application.

Démonstration, *proof-of-concept* : débogage et mise au point réduits.

Pas de souci d'industrialisation, de déploiement ou de mise à jour pour le moment.

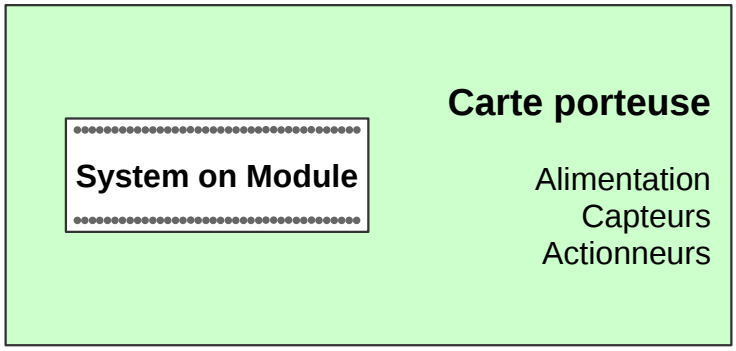


On teste éventuellement plusieurs SBC afin de déterminer le processeur le plus adapté.

Petite série – Startup

Computer-on-module (C.O.M.) - System-on-module (S.O.M)

Un module est une petite carte de dimension réduite contenant l'équivalent d'un ordinateur mono-carte (S.O.C, mémoire…) sans connecteurs.



La liaison avec la carte porteuse se fait soit par broches HE10 soit par collage CMS.

La conception de la carte porteuse doit être étudiée pour s'intégrer dans le boîtier final du projet.

C'est une phase qui fait souvent appel à du financement participatif (*crowd-funding*)

Développement logiciel

L'industrialisation du processus logiciel est cruciale.

Les systèmes de construction d'image (Buildroot, Yocto, etc.) sont préférés aux distributions.

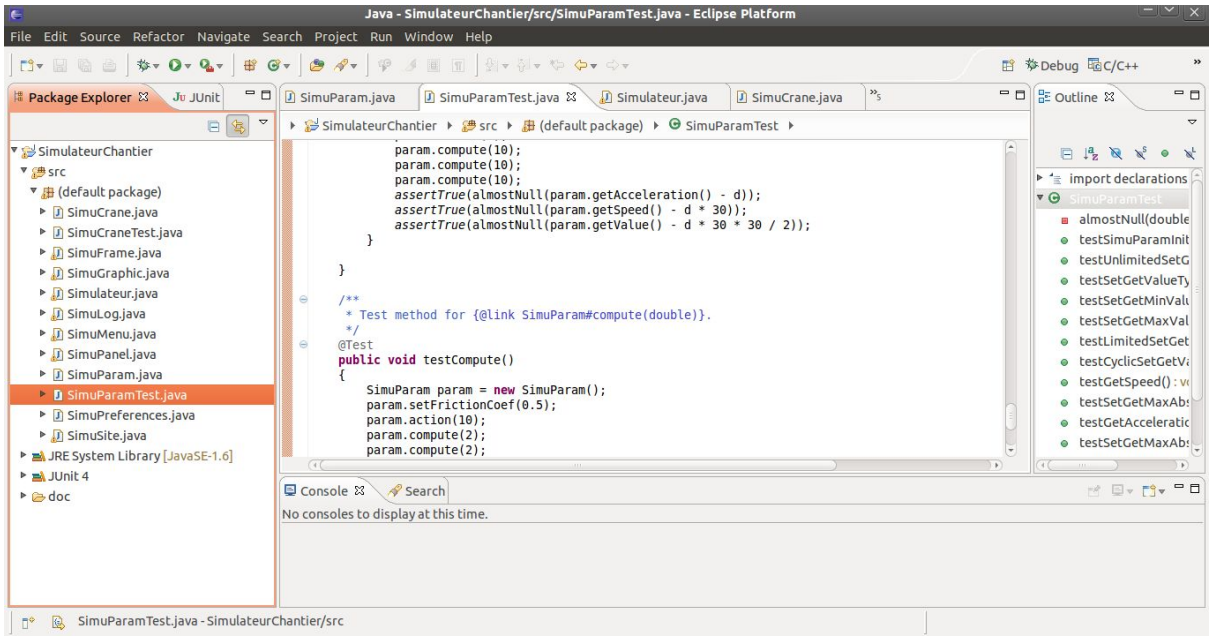
La configuration du noyau est optimisée (temps de boot, occupation mémoire, choix des drivers).

Il faut mettre au point des mécanismes de déploiement et de mise à jour.

Le logiciel doit être sécurisé pour être fiable (non-*brickable*) pour éviter les retours S.A.V.

C'est un bon moment pour mettre en place un environnement d'intégration continue, un système de gestion de version, etc.

I.D.E. - Éclipse



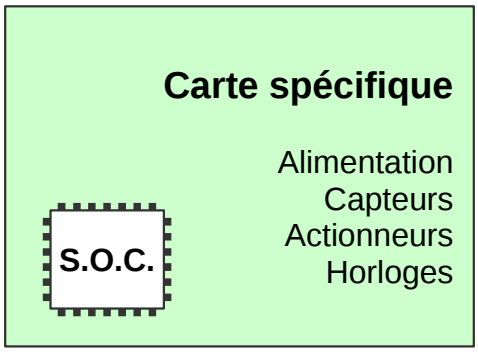
Éclipse est un environnement de développement intégré permettant de gérer des projet Java, C, C++, Python, etc.

Éclipse permet de réaliser du débogage local ou sur une cible distante.

Grande série – production industrielle

Intégration d'un *system-on-chip*

Rarement intéressant en dessous d'une dizaine de milliers d'unité.



Coûts importants de design, routage, banc de test, validation, etc.

Les frais de production sont avantageux.

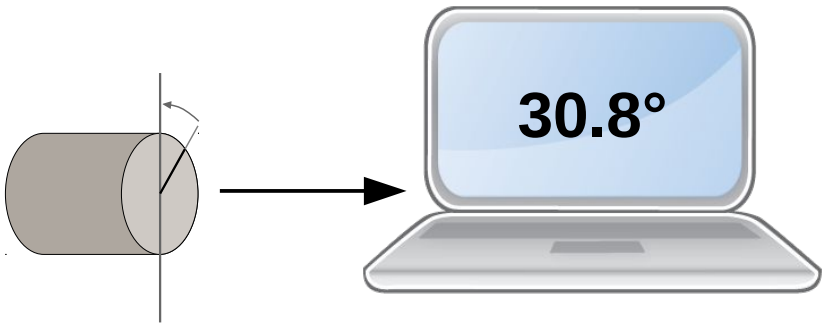
Externalisation de la conception : attention à la propriété intellectuelle

Étude de cas

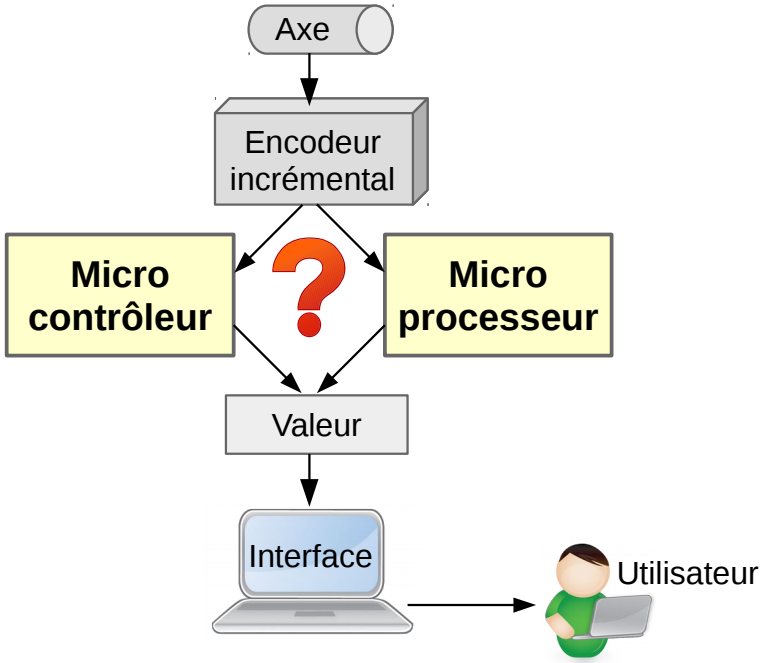
Problématique

Nous devons mesurer **l'angle de rotation d'un axe** par rapport à une position de repère.

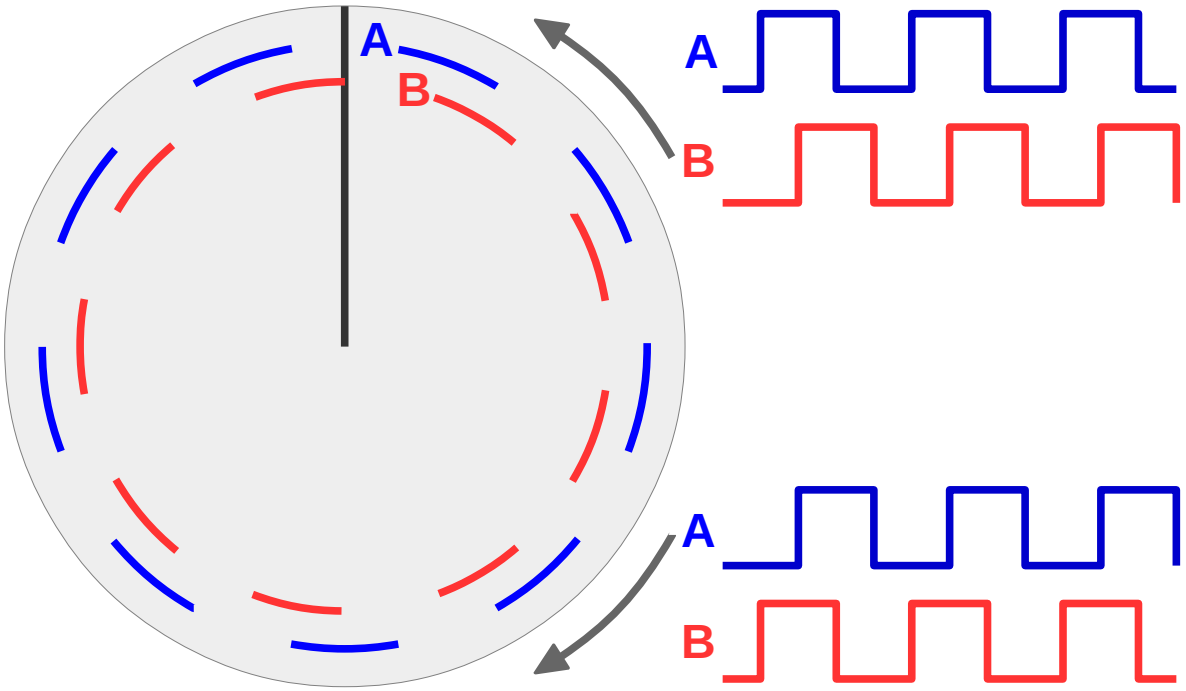
Nous voulons fournir à l'utilisateur la valeur de l'angle en **degrés et dixièmes**.



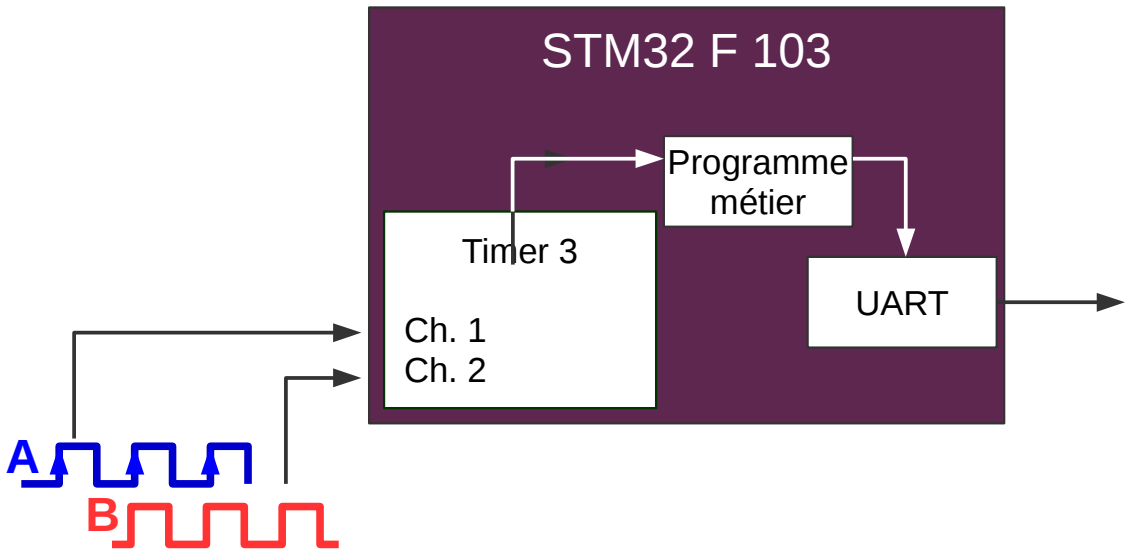
Architecture envisagée



Codeur incrémental de rotation



Système à microcontrôleur



Résultats

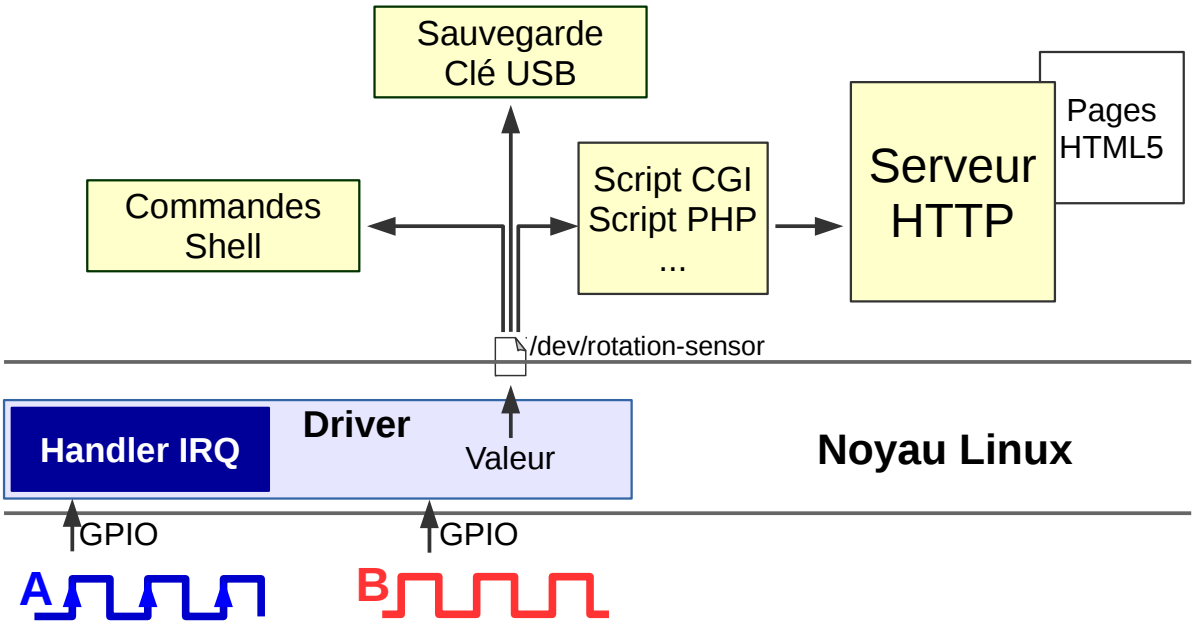
Validité des résultats, exactitude, précision..... 

Fiabilité du système, facilité d'industrialisation ..... 

Interface utilisateur, confort, ergonomie..... 

Évolutivité, souplesse d'adaptation à de nouvelles demandes..... 

Microprocesseur sous Linux



Résultats

Validité des résultats, exactitude, précision..... 

Cette architecture est néanmoins correcte pour des contraintes temporelles faibles

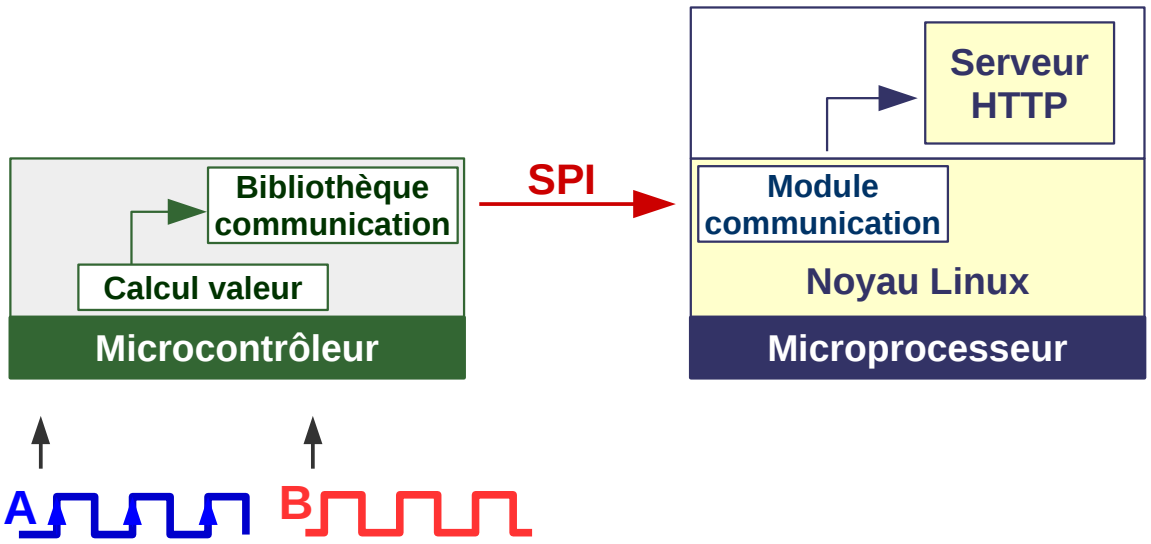
Fiabilité du système, facilité d'industrialisation ..... 

Le Raspberry Pi n'est pas industrialisable, mais il existe de nombreux systèmes acceptables.

Interface utilisateur, confort, ergonomie..... 

Évolutivité, souplesse d'adaptation à de nouvelles contraintes..... 

Architecture hybride



Projet LxMCU

Projet **libre** pour fournir une interface de communication bi-directionnelle entre **microcontrôleur** et **microprocesseur**, s'appuyant sur une liaison **SPI** (I²C à venir) en généralisant le principe des files de messages.

Côté microcontrôleur :

- Bibliothèque optimisée d'accès en lecture et écriture dans les files de messages.
- Notification lors de la réception d'un message.
- Licence LGPL : bibliothèque libre mais utilisable depuis code propriétaire.

Côté microprocesseur :

- Module kernel Linux avec interface « caractère » (`/dev/lxmcu0` … `/dev/lxmcu63`.)
- Communications synchrones ou asynchrones.
- Licence GPL : code libre dans le noyau Linux.

www.lxmcu.org

Résultats

Validité des résultats, exactitude, précision..... 

Fiabilité du système, facilité d'industrialisation ..... 

Interface utilisateur, confort, ergonomie..... 

Évolutivité, souplesse d'adaptation à de nouvelles demandes..... 

Conclusion

Le choix d'une architecture doit se faire en prenant en compte plusieurs facteurs :

- puissance de calcul et la quantité de mémoire nécessaires,
- le nombre et les types d'entrées-sorties utilisés,
- les 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 plate-forme utilisée pour le prototype ne sera pas la même que les premières séries ou la production en nombre.

Questions ?

N'hésitez pas à me contacter sur

christophe.blaess.fr

ou

www.logilin.fr