



1

Plan de Cours



Partie 1:

- **Architecture des systèmes embarqués & Linux embarqué**
 - Introduction aux systèmes embarqués
 - Caractéristiques des Systèmes Embarqués
 - Système d'exploitation temps réel
 - Linux Embarqué
 - Outils de développement de systèmes Linux embarqué
- **Partie Pratique:**
 - Linux Embarqué
 - Arduino
 - Raspberry
 - TinyOs

Partie 2:

- **Internet des Objets (IoT)** [Pr. El Brak Mohammed]

2

Contrat pédagogique

Horaire

- Cours : Mercredi de 9h à 12h15
- Projets/TP / rattrapage: Jeudi de 13h30 à 15h:15



Evaluation

- CC1: En début de Décembre 2020
- CC2: En Fin de Janvier 2021
- Note: %CC1+%CC2+%Projets+%Exposés+ %Absence
- Validation: Note >=12/20
- Assiduité/Rigueur/Autonomie sont des atouts considérés.

Déroulement

- Séances à distance (Hangout Meet)
- Activation de Caméra obligatoire pour suivre le cours
- Prise de note obligatoire, (Contrôlée et notée)
- Aucun support de cours ne sera communiqué,
- Délégué → Hub de communication

3



The collage includes the following book covers:

- Embedded Systems by Jonathan W. Valvano
- LOGICIEL LIBRE IoT by Various Authors
- Linux pour l'embarqué by Steve Heath
- Embedded Systems Design by Steve Heath
- Linux embarqué Comprendre, développer, élaborer by Steve Heath
- EMBEDDED SYSTEMS by Various Authors
- Objets connectés La nouvelle révolution industrielle by Various Authors

On the right side, there is a large blue graphic element containing the text "Systèmes Embarqués & IoT" and "Cycle Ingénieur: LSI2 Année universitaire : 2020/2021". Above this graphic is the Faculty of Sciences and Techniques logo for Tangier, featuring a globe and Arabic text "جامعة العلوم والتكنولوجيات - تانجيري".

4

Introduction

Systèmes Embarqués:

- Objets et appareils de la vie quotidienne,
- Facilite la vie ,
- Sauver des vies,



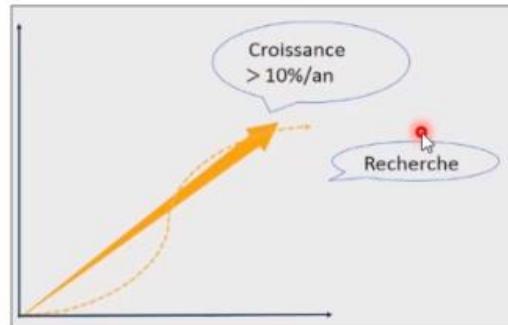
5

Introduction

Systèmes Embarqués:



•Secteur d'activité dynamique



6

Introduction

Systèmes Embarqués:



•Secteur d'activité dynamique

•Pluri et inter disciplinaire,

Développement logiciel



6

Introduction

Systèmes Embarqués:



•Secteur d'activité dynamique

•Pluri et inter disciplinaire,

Electronique



Automatique



6

Introduction

Systèmes Embarqués:



•Secteur d'activité dynamique

•Pluri et inter disciplinaire,

Mécatronique



6

Introduction

Systèmes Embarqués:



•Secteur d'activité dynamique

•Pluri et inter disciplinaire,

•Perspectives de nouvelles applications,

6

Introduction

Historique:



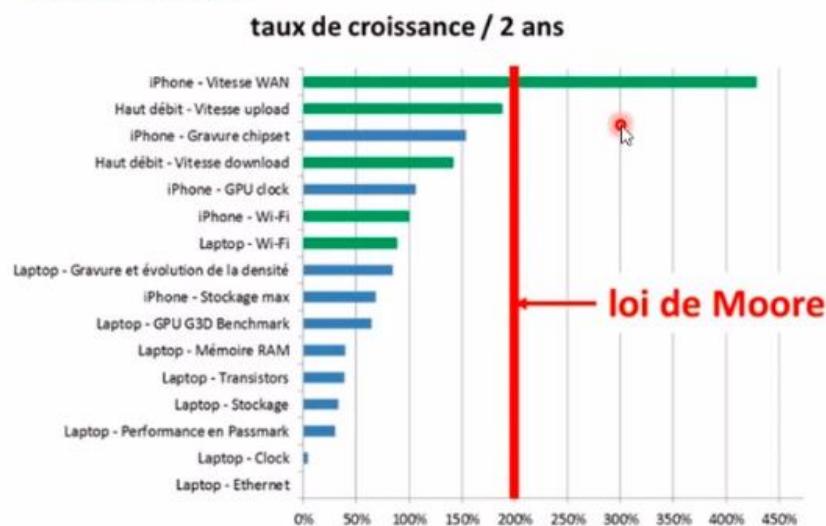
7

Introduction

Systèmes Embarqués: (Importance du Marché de l'embarqué)

Des lois empiriques qui sont bien vérifiées:

1- Loi de Moore :



8

Introduction

Systèmes Embarqués: (Importance du Marché de l'embarqué)

Des lois empiriques qui sont bien vérifiées:

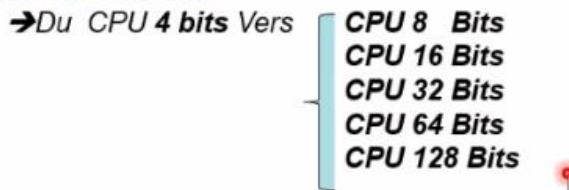
2- Loi de Joy :

• La puissance CPU des processeurs embarqués en MIPS double tous les 2 ans.

3- Loi de Ruge :

- besoin d'une Bande Passante de 0,3 à 1 Mb/s par MIPS.

Intégration sur silicium:



9

Introduction

Systèmes Embarqués: (Importance du Marché de l'embarqué)

Progrès dans l'Intégration sur silicium:



10

Introduction

Définition:

• On les appelle aussi des systèmes « **enfouis** ».

• Les systèmes embarqués, sont des systèmes **électroniques** et **informatiques autonomes**, construits pour effectuer des **tâches précises**.

• « **Embedded System** »: tout système conçu pour résoudre un problème ou une tâche spécifique mais n'est pas un ordinateur d'usage général.

• Capables :

- d'interagir avec leur **environnement**
- de gérer leurs ressources disponibles (*puissance de traitement, capacité de stockage, énergie, etc.*). 



11

Introduction

Définition:

- Est un système numérique.**
- Utilise généralement un processeur.**
- Exécute un logiciel dédié pour réaliser une fonctionnalité précise.**
- N'a pas réellement de clavier standard (BP, clavier matriciel...)**
- L'affichage est limité (écran LCD...) ou n'existe pas du tout.**
- N'exécute pas une application scientifique ou commerciale traditionnelle.**



12

Introduction

TYPES de Systèmes Embarqués:

- ❑ **Calcul général:**
 - Jeux,
- ❑ **Contrôle de système temps réel,**
 - Système de navigation aérien
- ❑ **Traitement de Signal**
 - Système des Radars
- ❑ **Transmission de l'information et commutation;**
 - Switch, ADSL, Routeur

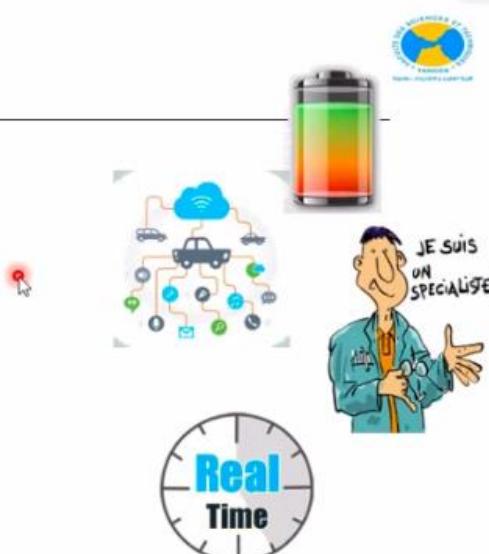


13

Introduction

Caractéristiques de Sys.Embarqués:

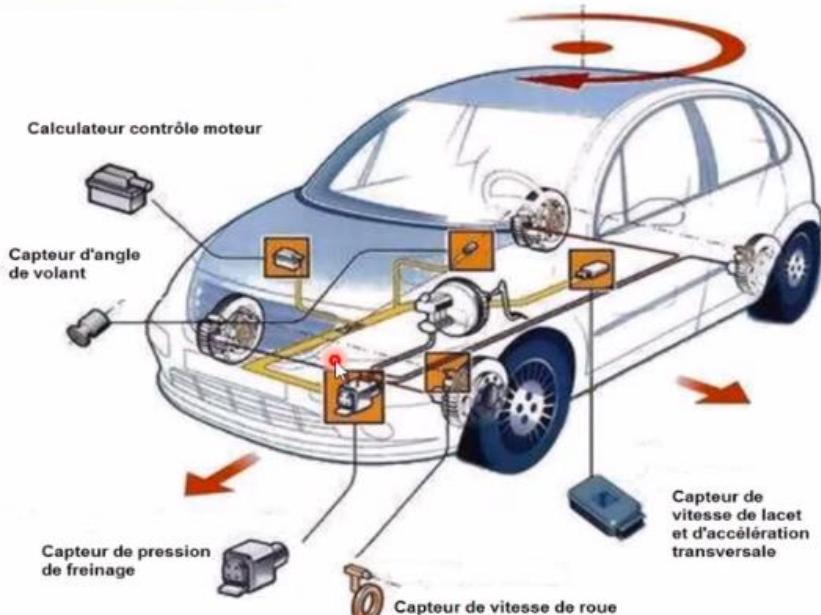
- Système Autonome,
- Connecté,
- Spécialisé,
- Temps réel,
- Eléments matériels et logiciels.



14

Introduction

Domaines d'application des systèmes embarqués:
Applications dotées de SE:



15

Introduction

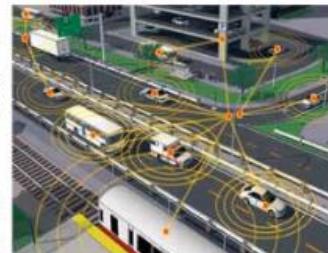
Domaines d'application des systèmes embarqués:
DOMAINE d'AUTOMOBILE

C'est dans ce domaine que s'applique une très grande quantité d'innovations, dont une majorité est fait des systèmes embarqués.

Expression des besoins et souhaits de l'automobiliste d'aujourd'hui :



- ✓ sécurité, fiabilité, communicabilité, intelligence, personnalisation, assistance, services, respect de l'environnement et (même) distraction,



- ✓ fonctions automatisées :Contrôle de vitesse et de distance entre véhicules, communication actualisée avec l'infrastructure routière... Autant d'exigences, autant de systèmes embarqués composites, interopérants, à fiabilité et capacité accrues.

16

Introduction

Domaines d'application des systèmes embarqués:
Applications dotées de SE:



Le **robot programmable** exécute le **programme** qui a été téléchargé dans sa **mémoire**. Il peut suivre une route, éviter des obstacles, jouer de la musique...

Le **robot aspirateur** est capable de nettoyer les sols et « **mémoriser** » la taille des pièces et obstacles rencontrés sur son passage afin de les éviter et d'optimiser les temps de parcours.



17

Introduction

Domaines d'application des systèmes embarqués:
Applications dotées de SE:



Le **drone** est capable de corriger sa position, de se **stabiliser** en restant en vol stationnaire. Il **exécute** aussi en temps réel les ordres envoyés par la radiocommande.

La **voiture sans conducteur** est capable de transporter des usagers sur n'importe quelle route en toute autonomie et sécurité.



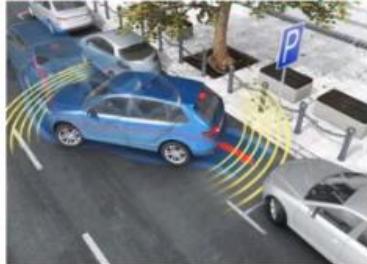
18

Introduction

Domaines d'application des systèmes embarqués:

Applications dotées de SE:

Stationnement automatique



Détecteur de pluie et essuie glace auto



Système d'aide à la conduite Détecteur d'objets



Système d'aide à la conduite Détection de ligne de route



19

Introduction

Domaines d'application des systèmes embarqués:

Applications dotées de SE:

Smart Home



20

Introduction



Domaines d'application des systèmes embarqués: Applications dotées de SE:



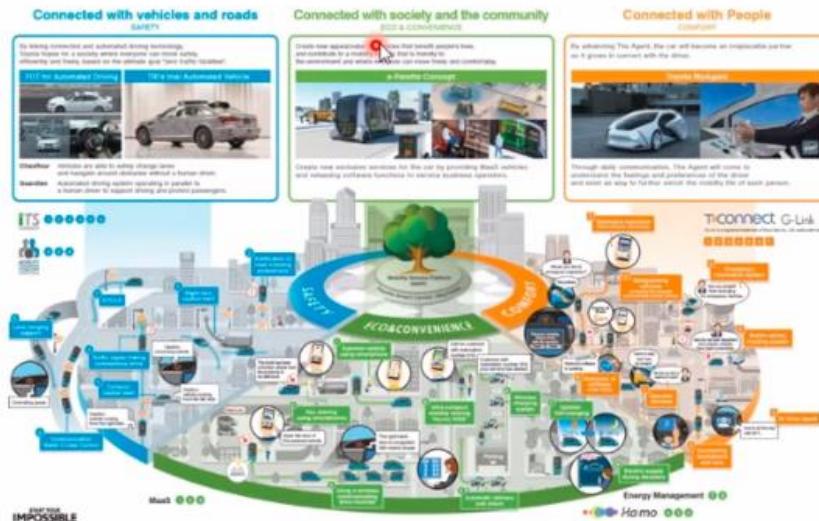
21

Introduction



Domaines d'application des systèmes embarqués:

Envisioning Smart Mobility Society in the Connected Future



22

Introduction

Quelques chiffres :

En moyenne, **1/3 du coût global d'un avion** est aujourd'hui lié aux systèmes embarqués, dont **40% en développement de logiciels**.

□ Environ 20% du coût d'une **automobile** vient de la conception et de la réalisation des **systèmes embarqués** ;

○ *Un véhicule peut regrouper aujourd'hui jusqu'à 70 ECU (Electronic Control Unit) ;*

○ *Le système embarqué a également un effet de levier important : le succès commercial d'une automobile dépend de plus en plus de la qualité des systèmes embarqués et de l'offre de services à l'utilisateur.*

□ La part du **logiciel embarqué dans le développement de nouveaux produits** et systèmes de distribution et de gestion d'énergie **dépasse les 15%**.



24

Introduction

Domaines d'application des systèmes embarqués:

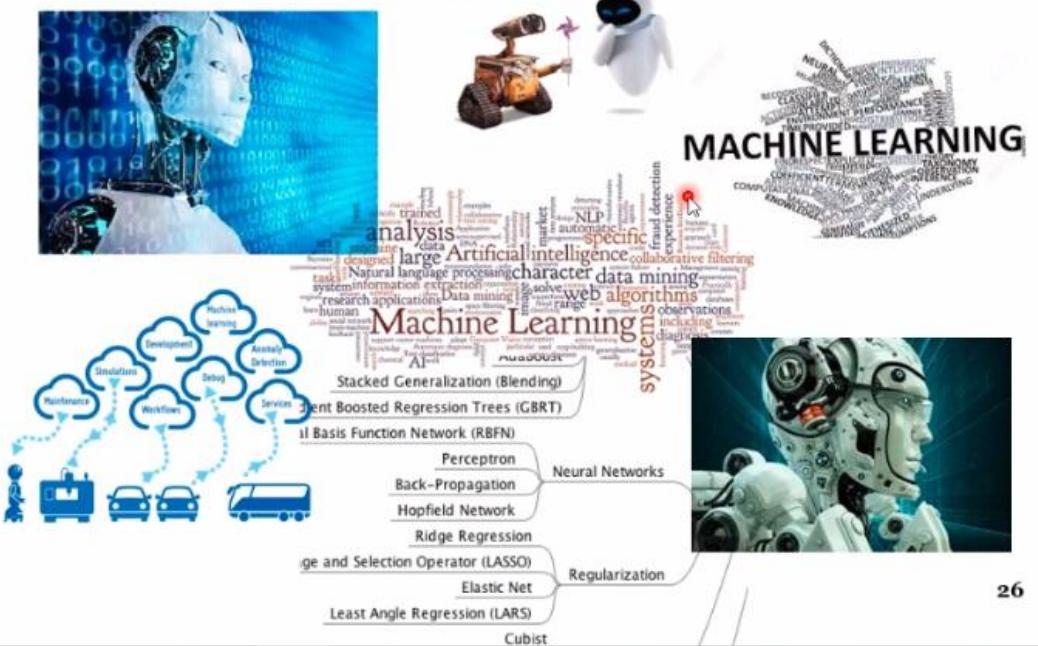
- | | |
|--|---|
| <ul style="list-style-type: none">✓ Téléphonie, Internet mobile✓ Avionique✓ Robotique✓ Automobile✓ Militaire✓ Jeux et loisirs | <ul style="list-style-type: none">✓ santé & sécurité✓ Domotique,✓ immeubles intelligents,✓ Villes intelligentes✓ Vêtements... |
|--|---|

25

Introduction



De l'embarqué vers de l'intelligence



Introduction



Caractéristiques des produits à base des systèmes embarqués:

- ❖ Un SE doit satisfaire à :
 - **Performances:**
 - Temps de réaction et d'exécution
 - Consommation énergétique
 - **Sûreté de fonctionnement**
 - Disponibilité,
 - Robustesse,
 - Fiabilité,
 - **Sécurité**
 - Sécurité des personnes et des biens,
 - Sécurité de l'information,
 - **Aspects économiques**
 - coût



Système Embarqué

L'association et l'intégration de *composants logiciels évolués, de puces électroniques puissantes et miniaturisées* permettent de:



- Développer plus d'intelligence,
- Assurer plus de sûreté,
- Rendre communicant tous les objets de notre quotidien, du plus petit au plus grand, du plus simple au plus complexe.



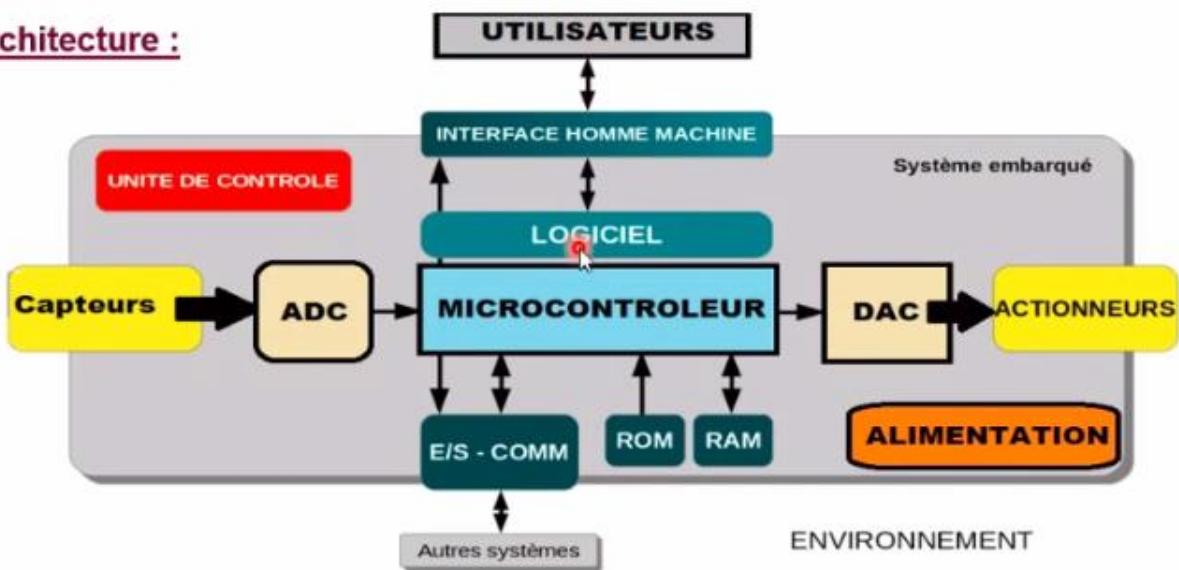
• L'émergence de « l'Internet des Objets » (Internet of things):

lien entre le monde du Web et celui des systèmes embarqués, amplifie de façon considérable cette révolution



Système Embarqué

Architecture :



Système Embarqué

Logiciel Embarqué:

Drivers:

- Gère les périphériques internes (MCU) et externes
- Gère les mémoires internes (MCU) et externes
- Gère les interfaces de communications



Système Embarqué

Rôle du Logiciel embarqué dans les Systèmes embarqués:

→ Le logiciel embarqué ne se limite pas aux seuls composants logiciels embarqués dans le produit final.

→ Il intervient également de façon majeure dans l'ensemble des étapes du **cycle de développement** d'un système embarqué,:

- Conception système,*
- Développements logiciels,*
- Tests,*
- Intégration,*
- Vérification et validation.*



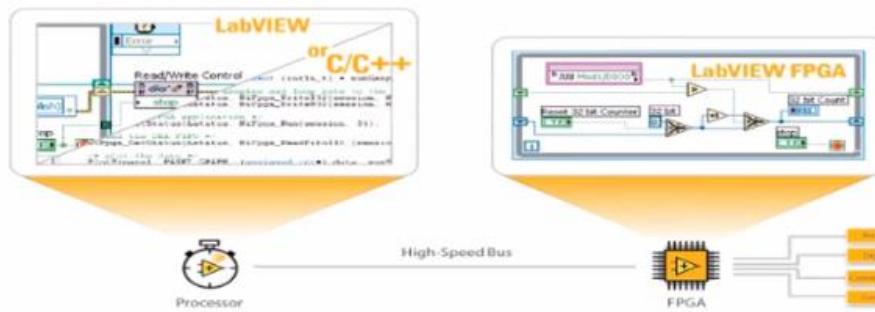
Système Embarqué

Développement de logiciel pour système embarqué:

Le développement de logiciels et matériels pour des systèmes à base de microcontrôleurs peut requérir différents outils :

- ❑ Des éditeurs,
- ❑ Des assembleurs,
- ❑ Des compilateurs

- ❑ Des “debuggers”,
- ❑ Des simulateurs,
- ❑ Des emulateurs

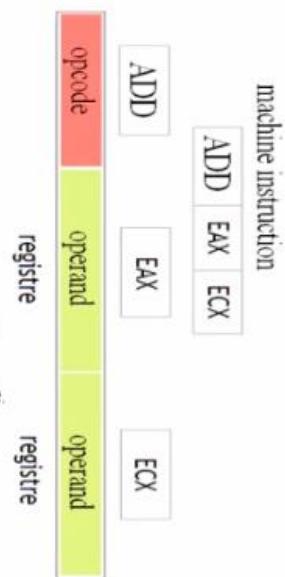


Système Embarqué

Ecriture du code pour le microcontrôleur:

L’écriture du code pour un microcontrôleur est réalisé à l’aide de langage relativement proche de la machine:

- ❑ En assembleur: jeu d’instruction directement compatible avec la machine.
 - **Avantages:** Compacité, rapidité pour l’exécution.
 - **Inconvénient:** spécifique à chaque type de machine connaissance du jeu d’instruction associé
- ❑ En langage de pas trop haut niveau; souvent en C



Système Embarqué

DES FONCTIONS CRITIQUES :

□ Un système est considéré comme **critique** dès lors que son dysfonctionnement, pour quelque cause que ce soit, pourrait avoir des **conséquences graves** sur la **sécurité de l'environnement**, des **personnes**, des **entreprises** ou des **biens**.



□ La **criticité** d'un système se traduit naturellement par un besoin fort de réduction du risque :

- **Réduction de la probabilité d'un dysfonctionnement** susceptible d'avoir de fortes conséquences,
- **Réduction des conséquences possibles d'un dysfonctionnement.**



34

Système Embarqué

Eléments de Criticité :

Trois éléments de criticité et donc de confiance dans les systèmes, doivent être distingués :

- **Sûreté de fonctionnement** (en anglais, « *dependability* ») : confiance portant sur la réalisation (dans tous les cas) de fonctions données ;
- **Sécurité des personnes et des biens** (« *safety* ») : confiance portant sur l'absence de conséquences graves, même en cas de panne ;
- **Sécurité de l'information** (« *security* ») : confiance dans la confidentialité et l'intégrité des données et dans l'impossibilité de détourner le système à des fins autres que celles prévues par ses concepteurs.



35

Système Embarqué

DES CONTRAINTES DE QUALITES DE SERVICE :

Les systèmes embarqués sont soumis à des limitations en termes de:

➔ **Contraints de fonctionnement:**

- ❑ volume, poids,
- ❑ puissance de calcul, mémoire disponible,
- ❑ alimentation et de consommation énergétique,
- ❑ Température, vibrations, radiations...



➔ **Contraints de sécurité,**

➔ **Contraints juridiques,**

➔ **Contraints environnementales,**

Développement durable, pollution...

➔ **Contraints de sociétales.**

Respect de la vie privée

36

Système Embarqué

DES LIMITATIONS TECHNIQUES ET ÉCONOMIQUES:

Les limitations peuvent être soit directement de nature technique, soit la conséquence de contraintes économiques:

- ❑ Coût de production, d'installation,
- ❑ Coût d'utilisation ou de maintenance du système embarqué.



38



Travail de Recherche

- Systèmes d'exploitation pour l'embarqué
- Linux Embarqué
 - Definition, Exemples, Création et installation
- Les outils de configuration de Systèmes d'exploitation pour l'embarqué,
 - ✓ Busy Box
 - Definition, Exemples, utilisation
 - ✓ Builtroot
 - Definition, Exemples, utilisation
- Chaine de compilation croisée
 - Definition, principe et fonctionnement



➔ A envoyer à l'adresse email du délégué.

➔ Délai: 11/11/2020 avant 00:00

39



Système Embarqué

Profil et débauches:



40



Système Embarqué

Temps de réponse du système:

C'est le **temps** passé entre la **présentation d'un ensemble d'entrées** à un système et la **réalisation du comportement requis**, dont la mise à disposition des sorties associées



Système temps réel:

C'est un système qui doit satisfaire des **contraintes sur le temps de réponse** déterminées au préalable ou qui est face à un risque de conséquences **critiques** ou non critiques..



Système Embarqué



Définitions:

- Un système temps réel est un système informatique soumis à des contraintes de temps.
- Un système est dit temps réel s'il doit s'exécuter suffisamment vite par rapport à la dynamique du procédé contrôlé. La notion relative de vitesse s'exprime par des contraintes temporelles.





Système Embarqué

Exemple : l'ESP (Electronic Stability Program) :

- **Capteurs concernés** : volant, roues.
- **Actionneurs** : injection moteur, frein.
- **Contrainte** : réaction sur les freins et l'injection suite à un coup de volant brusque en moins de 150 milliseconde => **temps de réponse ≤ 150 ms**.



43

Système Embarqué



Type de systèmes temps réel:

- **Contraintes temporelles relatives ou lâches** (temps réel mou : soft real-time) : les fautes temporelles sont tolérables *Ex. : jeux vidéo, applications multimédias, téléphonie mobile...*
- **Contraintes temporelles strictes ou dures** (temps réel dur : hard real-time) : les fautes temporelles ne sont pas tolérables *Ex. : fonctions critiques avionique, véhicules spatiaux, automobile, transport ferroviaire...*
- **Contraintes temporelles fermes** (temps réel ferme : firm real-time) : les fautes temporelles sont autorisées dans une certaine limite, par exemple une erreur toutes les trois exécutions au plus.
- **Systèmes multi-critiques** : les sous-systèmes composant le système sont caractérisés par des degrés de criticité.



44



Système Embarqué

Intégration de technologies:

Langages : Assembleur, C, C++, Java.

Java ME (Micro Edition), anciennement appelée J2ME : appareils trop limités pour faire tourner l'ensemble des librairies de Java SE Appareils mobiles Systèmes embarqués

Moniteurs/Noyaux temps réel : VxWorks, pSOS, Nucleus, ThreadX, Linux.

Systèmes d'exploitation Windows : Windows CE, Windows XPE.

Outils : Assembleur, compilateur, linker/locater, profiler (version commerciale ou Open source). Tornado, GNU, Eclipse, Microsoft Visual Studio, ARM Developers Suite ...

SDK/Toolkits/Libraries : Sous licence commerciale ou Open source.

46



Système Embarqué

Objectifs de Java ME :

- Écrire des programmes qui s'exécutent sur des téléphones
- Pas besoin d'être connecté en ligne
- Utilisation du processeur et de l'interface du téléphone

Applications

Jeux

Outils de calcul

Interface utilisateur pour un service (cartes pour téléphones...)



49



Système Embarqué

Ce que permet Java ME:

Calcul

Langage général

Limité uniquement par la vitesse du processeur et la mémoire

Graphique

Incluant la 3D sur les nouveaux téléphones

Connectivité

SMS, MMS, Bluetooth, HTTP

Autres fonctions

RFID, localisation...



50



Système Embarqué



Matériel nécessaire:

- La documentation (*datasheet*) sur les composants utilisés.
- L'outillage de base de l'électronicien
- Les outils d'analyse temporelle : **oscilloscope**, analyseur logique... Des composants de base (**résistances, condensateurs...**)
- Un **microprocesseur** ou un **microcontrôleur**
- Un **programmateur** de microcontrôleur
- Un émulateur in-circuit ou ICE (*In Circuit Emulator*).
- Le **débogage** du logiciel in situ (lecture/modification de registres, mémoires, périphériques...)
- la **programmation** de la mémoire **FLASH** des **microcontrôleurs**,

51



Système Embarqué

Pourquoi utiliser un système embarqué?

- Applications complexes avec multitude de fonctionnalités,
- Besoin de :
 - Système Multitâche,
 - Gestion de processus et de mémoire,
 - Communication entre processus,
 - Timers...



52



Linux Embarqué



53

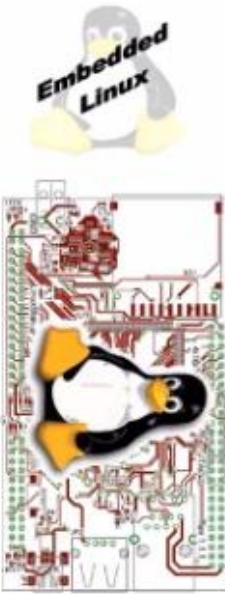
Système Embarqué

(Linux Embarqué)



Linux Embarqué:

- ❑ Le terme **Linux embarqué** apparaît en 1999.
- ❑ C'est un système d'exploitation basé sur **linux** et adapté à un **système embarqué**.
- ❑ Prédominance du **libre** dans le secteur industriel,
- ❑ Le matériel rejoint le logiciel;
➤(VHDL, Verilog) pour synthétiser et aussi tester les circuits numériques
- ❑ Temps Réel



54

Système Embarqué

(Linux Embarqué)



Linux Embarqué:

- ✓ Des versions de Linux embarqué pour :
 - ✓ Téléphones portables,
 - ✓ PDA,
 - ✓ set-top boxes, ...).
 - ✓ Client Cloud léger
 - ✓ Routeurs



55

Système Embarqué

(Linux Embarqué)



Linux Embarqué

→Fonctions:

- ✓ Traitement lié au fonctionnement de la machine.
- ✓ Communication avec un autre calculateur (Machine to Machine).
- ✓ Communication avec l'homme



→Caractéristiques:

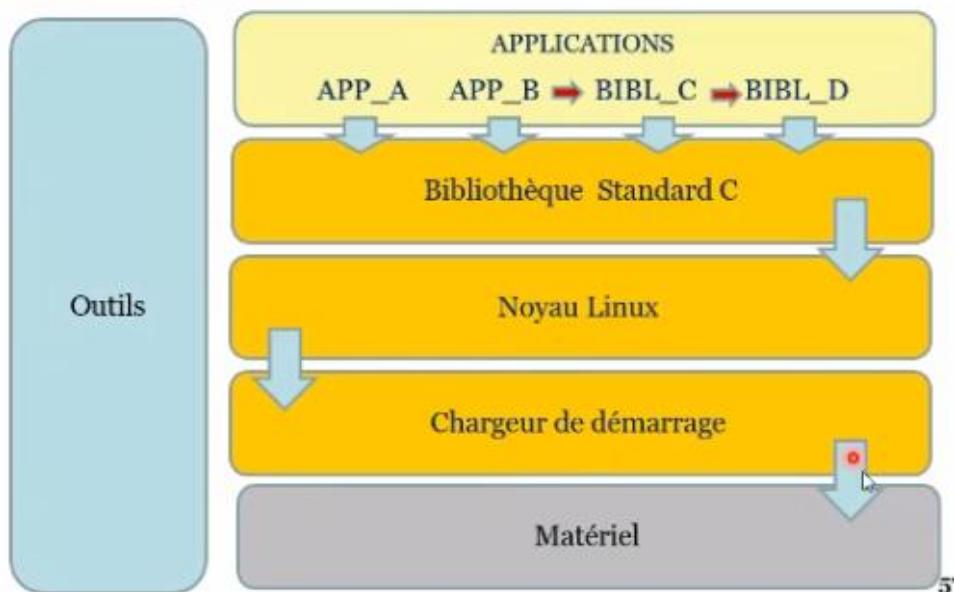
- ✓ Systèmes aux ressources **limitées**
- ✓ Disposent généralement de **peu de RAM**,
- ✓ Utilisent de la **mémoire flash** plutôt qu'un disque dur
- ✓ Dispose d'un **noyau temps réel** 



Système Embarqué

(Linux Embarqué)

Architecture de base:

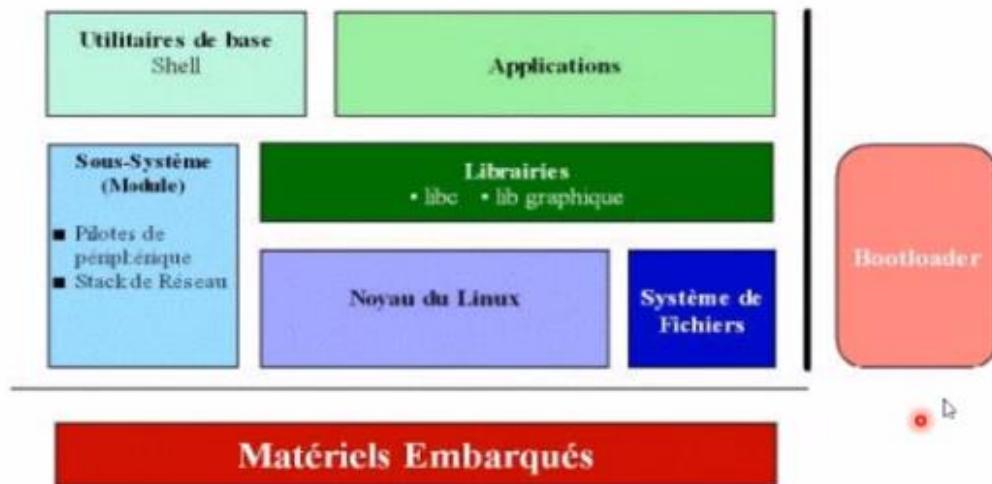




Système Embarqué

(Linux Embarqué)

Architecture fonctionnelle:



58

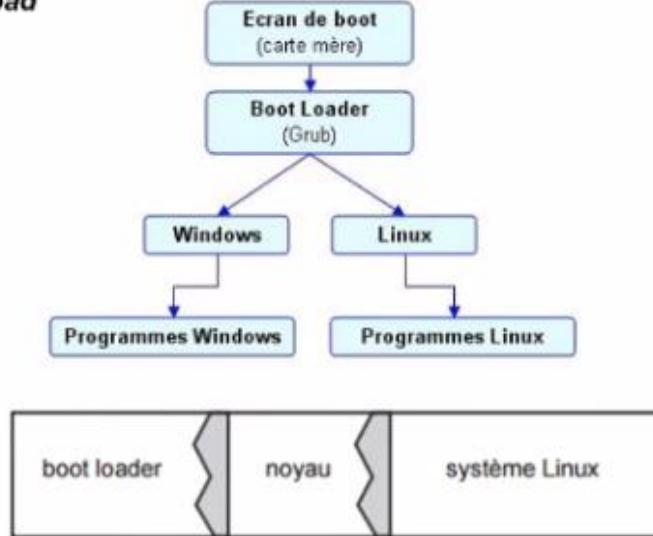


Système Embarqué

(Linux Embarqué)

Bootloaders:

Le démarrage d'un tel système nécessite un *chargeur d'amorçage* ou **bootload**



59



Système Embarqué

(Linux Embarqué)

Linux pour l'embarqué: Pourquoi?

-Les raisons généralement invoquées pour utiliser Linux comme OS embarqué sont les suivantes :

- **Fiabilité**
- **Faible coût**
- **Performances**
- **Portabilité**
- **Ouverture**

60



Système Embarqué

(Linux Embarqué)

Taille du Noyau Linux, (version 4.6):

- ➔ **Taille brute:** 730 MB
 - 53,600 fichiers,
 - approx 21,400,000 lignes)
 - **gzip** compressé tar archive: 130 MB
 - **xz** compressé tar archive: 85 MB

- ➔ **Linux Minimal V 3.17**
 - Compilé sur un noyau linux,
 - Avec un bootloader sur architecture ARM
 - Système de fichier : Ext2
 - **Compressé : 876 kb**
 - **Brut : 2.3 Mb**



61

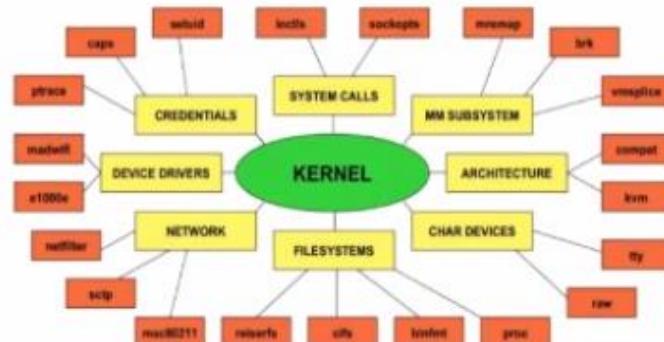
Système Embarqué

(Linux Embarqué)

Taille du Noyau Linux, (version 4.6)

Pourquoi ces sources de noyaux sont grandes?

- ✓ Inclusion de centaines de pilotes des matériaux,
- ✓ Plusieurs protocoles réseaux,
- ✓ Support de plusieurs architecture
 - Risc : Reduced Instruction set computing, (PowerPC, MIPS, ARM...)
 - Cisc : Complex Instruction set computing (x86, x64....)
 - EPIC: Explicitly Parallel Instruction Computing
 - VLIW:Very Long Instruction Word
- ✓ Fichiers système



62

Système Embarqué

(Linux Embarqué)

Taille du Noyau Linux, (version 4.6)

- | | |
|------------------------|-------------------|
| ▶ drivers/: 57.0% | ▶ firmware/: 0.6% |
| ▶ arch/: 16.3% | ▶ lib/: 0.5% |
| ▶ fs/: 5.5% | ▶ mm/: 0.5% |
| ▶ sound/: 4.4% | ▶ scripts/: 0.4% |
| ▶ net/: 4.3% | ▶ crypto/: 0.4% |
| ▶ include/: 3.5% | ▶ security/: 0.3% |
| ▶ Documentation/: 2.8% | ▶ block/: 0.1% |
| ▶ tools/: 1.3% | ▶ ... |
| ▶ kernel/: 1.2% | |

63



Système Embarqué

(Linux Embarqué)

Démarche et stratégie de construction de SE:

→ Du + vers le - :

- ❑ Commencer à partir d'un environnement de bureau complet sous GNU/Linux (Debian, Fedora...) et supprimer tous les éléments inutiles.
- ❑ Travail très complexe :
 - ✓ besoin de vérifier un très grand nombre de fichiers et de paquetages.
 - ✓ Besoin de comprendre l'(in)utilité de chaque fichier avant de le supprimer.
- ❑ Conserver des scripts et des fichiers de configuration inutilement complexes.
- ❑ Le résultat final est quand même trop gros car les outils et les bibliothèques standards sont utilisés. De plus, beaucoup de bibliothèques partagées sont nécessaires.

64



Système Embarqué

(Linux Embarqué)

Démarche et stratégie de construction de SE:

→ Du - vers le + :

- ❑ Commencer avec un **système de fichiers minimal**, voire vide, en ajoutant seulement les éléments qui vous sont nécessaires.
- ❑ Bien plus facile à réaliser:
 - ❑ temps consacré au besoin.
 - ❑ Plus facile à contrôler et à maintenir : développement d'une compréhension des outils à utiliser.
 - ❑ besoin de scripts de configuration basiques.
- ❑ Le résultat final peut être extrêmement petit, d'autant plus qu'on utilise des outils légers.

65



Système Embarqué

(Linux Embarqué)

Avantages

- ✓ "Code Source " disponible et gratuit .
- ✓ Beaucoup de pilotages et outils de développement disponibles.
- ✓ Plein d'application .
- ✓ Support de réseau .
- ✓ Fiable et robuste.

Inconvénients:

- ✓ Linux n'est pas un micro-noyau.
- ✓ Aucune société n'est à l'origine du développement de GNU/Linux.

66



Système Embarqué

(Linux Embarqué)

Distributions pour l'embarqué:

- ✓ Red hat
- ✓ Montavista
- ✓ Mandriva
- ✓ Android
- ✓ µClinix
- ✓ ...



montavista



Mandriva



µClinix

67



Système Embarqué

[\(Linux Embarqué \)](#)

Noyau Linux:

✓ Version de mon kernel courant:

- \$ uname -r

✓ Localisation de fichiers:

- Les **sources du noyau** sont situés dans **/usr/src/linux-<version>/**.
- Lien symbolique **/usr/src/linux/** qui pointe vers ce répertoire.
- Le **noyau** est situé dans le répertoire **/boot/**.
- Les **modules noyau** sont situés dans **/lib/modules/<version>/**.

✓ Composition du noyau:

- **vmlinuz** est l'image du noyau - le fichier qui contient le noyau.
- **System.map** est le fichier qui contient les symboles noyau requis par les modules pour assurer le lancement avec succès des fonctions du noyau.
- **initrd** - initial ram disk - est un fichier qui charge les drivers compilés en modules nécessaire au démarrage du noyau.

68

Système Embarqué

[\(Linux Embarqué \)](#)



Sources de noyau Linux:

- ✓ Vanilla
- ✓ Open SUSE



✓ Noyaux pour l'embarqué:

- VxWorks
- QNX
- XNU
- Minix
- LinuxWorks



69

Système Embarqué

(Linux Embarqué)



Configuration du noyau:

The screenshot shows the "Linux Kernel Configuration" window. On the left is a vertical list of menu items: Code maturity level options, Loadable module support (highlighted with a red circle), Processor type and features, General setup, Memory Technology Devices (MTD), Parallel port support, Plug and Play configuration, Block devices, Multi-device support (RAID and LVM), Networking options, Telephony Support, and ATA/IDE/MFM/RLL support. To the right are several groups of options: SCSI support, Fusion MPT device support, IDE/ATAPI (RawDevice) support (EXPERIMENTAL), I2O device support, Network device support, Amateur Radio support, SATA (AHCI) support, ISDN subsystem, OEM CD-ROM drivers (not SCSI, not IDE), Input core support, Character devices, Multimedia devices, File systems, Console drivers, Sound, USB support, Bluetooth support, Kernel hacking, Library routines, Save and Exit, Quit Without Saving, Load Configuration from File, and Store Configuration to File.

70

Système Embarqué

(Linux Embarqué)



Configuration du noyau:

- ✓ **make config** : programme en **mode texte** qui énumère toutes les options et demande d'entrer son choix.
- ✓ **make menuconfig** : utilitaire en **mode texte** qui permet une navigation plus aisée dans la configuration
- ✓ **make gconfig** : **outil graphique** basé sur [GTK+](#)
- ✓ **make xconfig** : **outil graphique** basé sur [Qt](#)
- ✓ **make defconfig** : outil permettant de récupérer les paramètres de configuration par défaut du noyau. Ces paramètres sont donnés par les développeurs du noyau à chaque nouvelle sortie du noyau.

70



Système Embarqué

(Linux Embarqué)

Compilation du noyau:

- ✓ Se fait par la commande **make**
- ✓ Opération peut être assez longue
- ✓ "j" : Option de réduction de temps de compilation
NB: L'option -j définit le nombre maximal de tâche en parallèles qui peuvent être exécutées par le makefile
Nombre de processeur ou core?
`$ export NBCOEUR=$((grep -c processor /proc/cpuinfo)+1))`
- ✓ **make install** (en mode root) → Installation de l'image du noyau.
 - Pas nécessaire lors de compilation pour Sys.Embarqué
- ✓ **make modules_install** (en mode root) → Installation des modules du noyau.
 - S'effectue dans « /lib/modules/<version>/ »

71



Système Embarqué

(Linux Embarqué)

Etapes de Compilation du noyau:

1) Récupération à partir du site web officiel (kernel.org)

The Linux Kernel Archives



About Contact us FAQ Releases Signatures Site news

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Release
5.9.8

maintain	5.10-rc3	2020-11-09	[tarball]	[patch]	[inc, patch]	[view diff]	[browse]	[change log]
stable	5.9.8	2020-11-10	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]
stable	5.8.18 [FOU]	2020-11-08	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]
longterm	5.4.77	2020-11-10	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]
longterm	4.19.157	2020-11-10	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]
longterm	4.34.206	2020-11-10	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]
longterm	4.9.243	2020-11-10	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]
longterm	4.4.243	2020-11-10	[tarball]	[pgpl]	[patch]	[inc, patch]	[view diff]	[browse]

72

Système Embarqué

(Linux Embarqué)



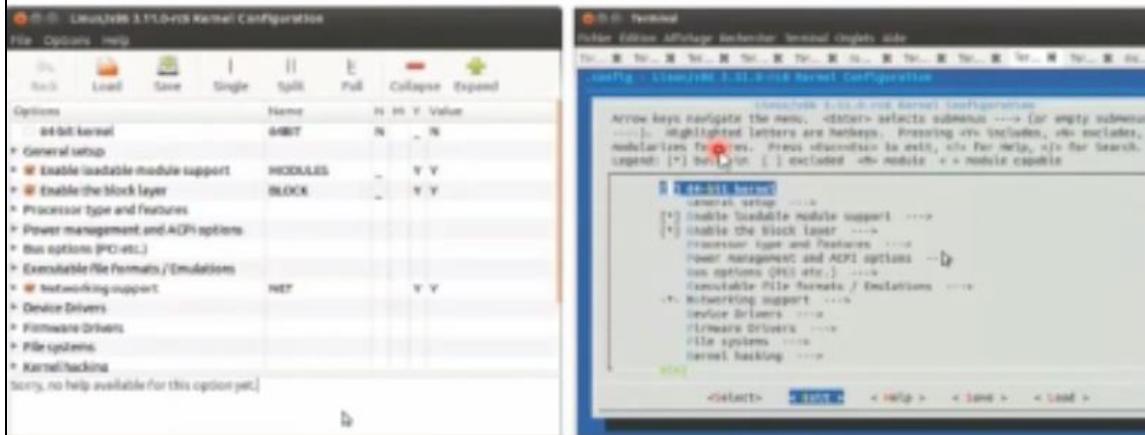
Etapes de Compilation du noyau:

2) Vérifier que l'intégrité d'une version téléchargé:

Validation par clef pgp publiée dans le site [le site kernel.org](http://kernel.org)

([S wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.6.6.tar.sign](https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.6.6.tar.sign))

3) Définir une nouvelle configuration:



Système Embarqué

(Linux Embarqué)

Etapes de Compilation du noyau:

4) Définir une nouvelle configuration:

\$ sudo apt-get install libncurses5-dev qt4-dev-tools libglade2-dev libglade2-0 libgtk2.0-0 libgtk2.0-dev libglib2.0-0 libglib2.0-dev

5) Génération du noyau:

→ Elle s'effectue en deux phases :

- la partie statique du noyau (**vmlinuz**, **bzimage**, ...);
- la partie dynamique des drivers (*.ko)

\$ sudo make scripts



```
HOSTCC scripts/basic/fixdep  
HOSTCC scripts/kconfig/conf.o  
HOSTCC scripts/kconfig/conf.o  
scripts/kconfig/conf -silentoldconfig Kconfig  
HOSTCC scripts/genksyms/genksyms.o  
HOSTCC scripts/genksyms/lex.lex.o  
HOSTCC scripts/genksyms/parse.tab.o  
HOSTLD scripts/genksyms/genksyms  
CC scripts/mod/empty.o  
HOSTCC scripts/mod/mk_elfconfig  
MKELF scripts/mod/elfconfig.h  
HOSTCC scripts/mod/title/alias.o  
HOSTCC scripts/mod/modpost.o  
HOSTCC scripts/mod/modsumversion.o  
HOSTLD scripts/mod/modpost  
HOSTCC scripts/seLinux/genheaders/genheaders  
compilation terminée
```



Système Embarqué

(Linux Embarqué)

Etapes de Compilation du noyau:

5) Génération du noyau:

`make` : pour lancer la génération en prenant en compte la configuration définie dans le fichier `.config`.

Génération de:

- `vmlinux` (Format ELF, non bootable)
- `arch/<arch>/boot/*Image` (image bootable)
 - bzImage pour une architecture x86
 - zImage pour une architecture ARM
- `arch/<arch>/boot/dts/*.dtb` (arborescence des composants compilés)
- Modules du noyau (fichiers `.ko`)

75



Système Embarqué

(Linux Embarqué)

Etapes de Compilation du noyau:

5) Test du nouveau noyau:

- Création d'un fichier vide de 320 Ko:

```
dd if=/dev/zero of=rootfs.img bs=320k count=1
```

- Formatage du fichier en ext2

```
mkfs.ext2 i 1024 F rootfs.img
```

- Intéressant de pouvoir le tester dans une machine virtuelle,

 QEMU permet la virtualisation sans émulation, si le système invité utilise le même processeur que le système hôte, ou bien d'émuler les architectures des processeurs **x86, ARM, PowerPC, Sparc, MIPS**

- <https://www.qemu.org/>

76



Système Embarqué

(Linux Embarqué)

TP 1:

En suivant les étapes citées ou encore à l'aide des tutoriaux:

- Configurer et compiler un nouveau noyau linux,**
 - Quel est le temps de compilation du noyau, ?
 - Comment améliorer ce temps de compilation?
- Donner sa taille brute et compressée,**
- Tester le noyau compilé sur QEMU,**

77



Système Embarqué

(Linux Embarqué)

Processus de développement de SE:

Environnement:

- Plate forme matérielle** qui va accueillir l'OS et les applicatifs embarqués
- Plate forme de développement** sur laquelle sont mis au point les logicielles de la cible

78



Système Embarqué

(Linux Embarqué)

Chaîne de compilation:

Outils de développement:

- Des compilateurs,
- Des interpréteurs,
- Des linkers,
- Des IDE (environnement de développement intégré),
- Des bibliothèques,
- D'un ensemble d'outils de la manipulation des fichiers binaires
- D'autres outils de développement.



79

Système Embarqué

(Linux Embarqué)



Construction d'un SE:

Réduction du système:

- 1) Optimisation de la procédure de démarrage du système (*services*),
- 2) Réduction du nombre de commandes disponibles et des bibliothèques partagées,
- 3) Optimisation du noyau en fonction des fonctionnalités nécessaires (*pilotes de périphériques, réseau ou non, etc...*)
- 4) Réduction et simplification du nombre de fichiers et répertoires de configuration (*/etc* contenant la majorité des fichiers de config)
- 5) Suppression du système de swap



Applicatif

Commandes + Lib

Kernel

Pilotes de Périphériques

Matériel

Structure d'un système LINUX simplifiée

80

Système Embarqué

[\(Linux Embarqué \)](#)



Construction d'un SE:

Terminologie et processus de dev.Linux embarqué:

- 1) Chaînes de **compilation croisée**,
- 2) **bibliothèques standard C pour l'embarqué.**
- 3) Chargeurs de démarrage (**bootloaders**).
- 4) **Configuration et compilation du noyau Linux.**
- 5) Applications et **bibliothèques légères** pour **systèmes embarqués**
- 6) **Systèmes de fichiers** traditionnels et spécialisés pour **stockage flash**.
- 7) **Outils de développement** de systèmes embarqués **Linux**.
- 8) **Développement et mise au point d'applications** sur le système embarqué.
- 9) **Contraintes** temps-réel et Linux embarqué

81

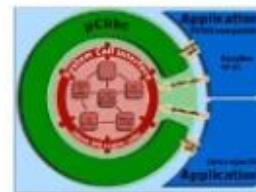
Système Embarqué

[\(Linux Embarqué \)](#)



Utilitaire pour la création d'un système Linux minimaliste à partir de zéro:

- 1) **BusyBox**
- 2) **BuildRoot**
- 3) **μClibc**
- 4) **OpenEmbedded**
- 5) **Crosstool**



82

Système Embarqué

(Linux Embarqué)



Exemple de Carte doté de Sys.Embarqué:

BeagleBone Black

<http://beagleboard.org/black>



What is BeagleBone Black?

BeagleBone Black is a low-cost, community-supported development platform for developers and hobbyists. Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.

Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 40 pin headers

Software Compatibility

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ BoneScript library
- plus much more

83

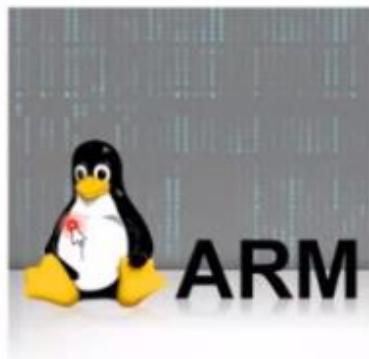
Système Embarqué

(Linux Embarqué)



Exemple de Carte doté de Sys.Embarqué:

LINUX EMBARQUÉ POUR LES CAMÉRAS INDUSTRIELLES



<https://fr.ids-imaging.com/linux-embedded-software.html>

84



Système Embarqué

(Linux Embarqué)

Exemple de Carte doté de Sys.Embarqué:
ARDUINO

WWW.arduino.com

The screenshot shows the official Arduino website at <https://www.arduino.cc>. The header features the Arduino logo and a Genuino logo. Navigation links include Home, Buy, Download, Products, Learning, Forum, Support, and Blog. A search bar is also present. The main content area includes a "WHAT IS ARDUINO?" section with an image of the Arduino Uno board, a "LED STRIPS INSTALLATION BEYOND XMAS" video thumbnail, and a "MICROSOFT & ARDUINO" challenge banner for the world's largest maker challenge.

85

Système Embarqué

(Linux Embarqué)

Exemple de Carte doté de Sys.Embarqué:
ARDUINO

WWW.arduino.com



Système Embarqué

■ L'Arduino est une carte basée sur un microcontrôleur (mini-ordinateur) Atmel ATMEGA8 ou ATMEGA168,



- 1 Ko de mémoire vive
- 8Ko de mémoire flash pour stocker ses programmes.
- 13 entrées ou sorties numériques
- 3 sorties analogiques
- 6 entrées analogiques



86

Système Embarqué

■ L'Arduino doté de système d'exploitation temps réel, avec système multi-tâches:



■ **DuinOS:**

- ✓ version basée sur FreeRTOS,
- ✓ Licence est dérivée de la GPL (GNU)
- ✓ prévu pour s'intégrer et fonctionner avec l'IDE d'Arduino,



■ **ViperOS**

- ✓ Version Linux
- ✓ Version d'Ubuntu



87

Système Embarqué

(Linux Embarqué)



Travaux Pratiques

IDE en ligne: <https://www.arduino.cc/en/Guide/HomePage>



```
PROGRAMME DE FONCTIONNEMENT 1.0.1  
File Edit Options View Sketch Hardware Web Tools Help  
C:\Program Files (x86)\ARDUINO\...\ARDUINO  
Project Explorer  
C:\Program Files (x86)\ARDUINO\...\ARDUINO\...\ARDUINO  
Terminal.ino  
Object Explorer  
void setup()  
{  
    // Set baudrate to 115200 baud  
    Serial.begin(115200);  
}  
  
void loop()  
{  
    // Read the analog input A0  
    Serial.print(analogRead(A0));  
    Serial.print(" - ");  
    Serial.print(c++);  
    delay(2000);  
    c++;  
  
    // Save data after 50 measurements into the logfile  
    if(c==50)  
    {  
        Serial.println("#$AV#"+logfile.txt);  
    }  
}  
  
Arduino Messages Search Results  
Verify code please wait.  
Last verify at 09:08:37  
Sketch uses 2.621 kilobytes (8%) of program storage space. Maximum is 32.256 kilobytes.  
Data Lock Disabled Line 26 of 51 Comport (COM3) Arduino Mega 2560 Board Arduino Uno  
88
```

Système Embarqué

(Linux Embarqué)



Travaux Pratiques

IDE en ligne: <https://codebender.cc/>



```
codebender (beta) Search  
Logged In as codebender Log Out  
Blink Example Clone Project  
Blink Example.ino +  
Delete Download Revert Save  
Cloud Section  
IP Address Monitor Flash  
Arduino Uno Verify Code  
/dev/cu.Microcontroller-WirelessAP USB Flash  
Speed: 9600 Open Serial Monitor  
Verification failed.  
Blink Example.ino:17:24: error: expected ';' after expression  
digitalWrite(C13, HIGH); // set the LED on  
^  
|  
1 error generated.  
Number of lines: 21  
89
```

The screenshot shows the CodeBender IDE interface with the 'Blink Example' sketch loaded. The code is as follows:

```
/*  
 * Blink  
 * Turns on an LED on for one second, then off for one second, repeatedly.  
 *  
 * This example code is in the public domain.  
 */  
  
void setup()  
{  
    // initialize the digital pin as an output:  
    // Pin 13 has an LED connected on most Arduino boards:  
    pinMode(C13, OUTPUT);  
}  
  
void loop()  
{  
    digitalWrite(C13, HIGH); // set the LED on  
    delay(1000); // wait for a second  
    digitalWrite(C13, LOW); // set the LED off  
    delay(1000); // wait for a second  
}
```

Système Embarqué

(Linux Embarqué)



Exemple de Carte doté de Sys.Embarqué:

RASPBERRY PI 3 MODEL B

[Raspberry](#)



90

Système Embarqué

(Linux Embarqué)



➔ Raspberry pi



- ❑ Carte mère de mini-ordinateur de la taille d'une carte de crédit ,
- ❑ Très bas coût (environ 30 € pour le modèle b sorti en février 2012),
- ❑ Initialement destiné à l'initiation de la programmation informatique surtout en terme d'embarqué,
- ❑ Composition minimale:
 - une carte SD ; *Disque Dur*
 - un adaptateur secteur (5 V – 1 A) pour alimenter la carte via la prise micro-USB ;
 - un adaptateur HDMI-VGA ou HDMI-DVI ;
 - un câble Ethernet RJ45.





Système Embarqué

(Linux Embarqué)

→ Préparation de la carte Raspberry pi



Raspberry Pi

□ Installer le système d'exploitation dans la carte SD,

□ Raspberry Pi fonctionner sous le système d'exploitation **GNU Linux**

□ Distributions Linux adaptées:

- Raspbian, Arch, Pidora,...



archlinux. pidora

92

Système Embarqué

(Linux Embarqué)



Raspberry Pi

→ Applications de Raspberry pi

1. Un serveur WEB
2. Un serveur BitTorrent
3. Un serveur VPN
4. Un ordinateur de bureau
5. Un serveur de mail De la domotique
6. Un serveur Webcam
7. Une tablette tactile
8. Un trackeur GPS
9. Un moniteur de température et d'humidité
10. Un serveur VOIP (Asterix)
11. Un lecteur RFID
12. Une caméra pour surveillance ou contrôle d'accès
13. Un lecteur RFID
14. Applications de domotique
15. Applications IoT

✓ <http://www.place4geek.com/blog/2014/09/plus-de-70-idees-de-projets-pour-votre-raspberry-pi/>

93

Buildroot



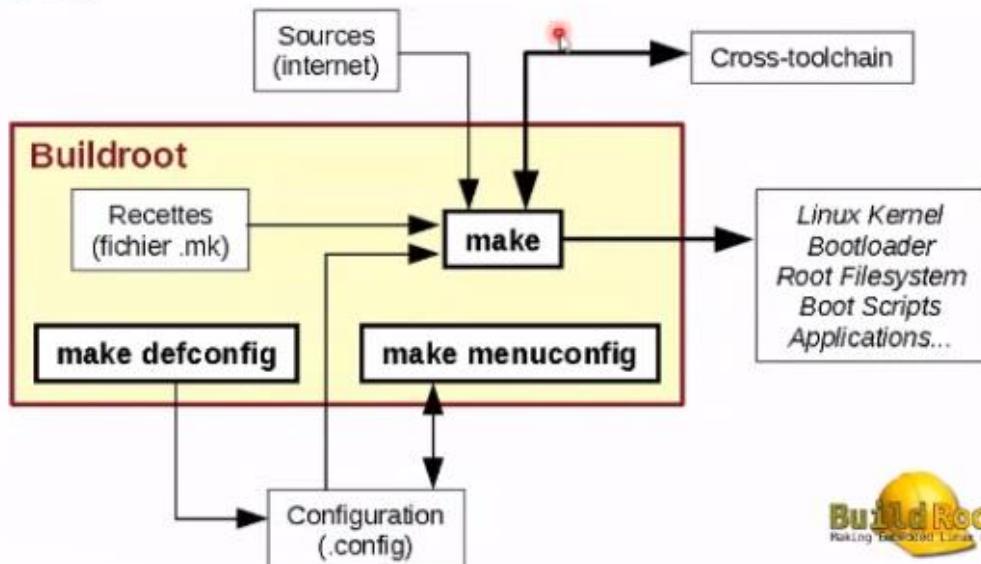
Références

<http://buildroot.org/downloads/>
<https://buildroot.org/downloads/manual/manual.pdf>
<https://www.youtube.com/watch?v=sH5xt4qbJbk>
<https://www.youtube.com/watch?v=1PfthHCfudY>

94

Système Embarqué [\(Linux Embarqué \)](#)

Buildroot est un ensemble de scripts et de fichiers de configuration permettant la construction complète d'un système Linux pour une cible embarquée. Il télécharge automatiquement les paquetages nécessaires pour la compilation et l'installation.



Système Embarqué

(Linux Embarqué)



→A propos de Buildroot:



- ❑ Buildroot est un outil qui simplifie et automatise le processus de construction d'un système Linux complet pour un système embarqué,
- ❑ Utilise Compilation croisée.
- ❑ Générer
 - Une chaîne cross-compilation,
 - Un système de fichiers racine en quelques minutes
 - Une image de noyau Linux
 - Un Bootloader.
- ❑ Buildroot est utile principalement pour les personnes travaillant avec des systèmes embarqués.
- ❑ Processeurs PowerPC, MIPS, ARM, etc.

96

Système Embarqué

(Linux Embarqué)



→Packaging obligatoire:

(dépend de chaque distribution)



- ❑ which
- ❑ sed
- ❑ make (version 3.81 or any later)
- ❑ binutils
- ❑ build-essential
 - ↳ *(only for Debian based systems)*
- ❑ gcc (version 2.95 or any later)
- ❑ rsync
- ❑ g++ (version 2.95 or any later)
- ❑ bash
- ❑ patch
- ❑ gzip
- ❑ bzip2
- ❑ perl (version 5.8.7 or any later)
- ❑ tar
- ❑ cpio
- ❑ python (version 2.6 or any later)
- ❑ unzip

97

Système Embarqué

(Linux Embarqué)



➔**Packtage optionel:**
(dépends de chaque distribution)

❑ Interfaces de configuration:

- curses5 → menuconfig interface
- qt4 → xconfig interface
- glib2, gtk2 ,glade2 → gconfig interface

❑ Packages Java:

- The javac compiler
- The jar tool

❑ Outils de génération graphique

- graphviz → graph-depend
- python-matplotlib → graph-build

98

Système Embarqué

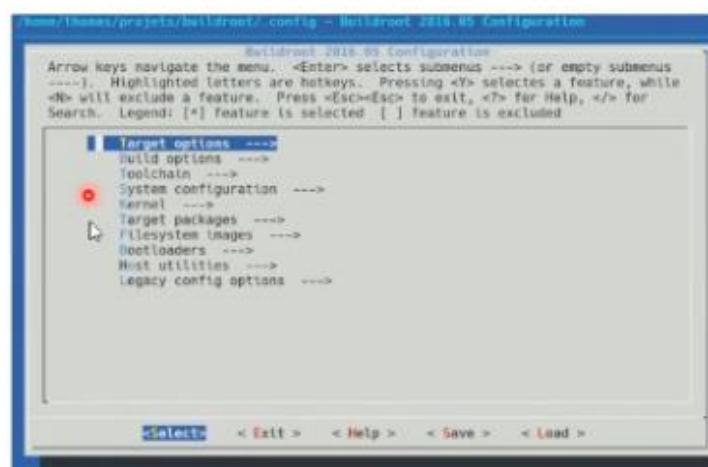
(Linux Embarqué)



➔**Utilisation de buildroot:**

❑ Un choix d'interfaces de configuration:

- make menuconfig
- make nconfig
- make xconfig
- make gconfig



Système Embarqué

(Linux Embarqué)



➔ Utilisation de buildroot:



❑ Exécution:

- \$ make

❑ Résultat d'exécution:

- Une ou plusieurs images du système de fichiers racine, sous différents formats
- Une image du noyau, éventuellement un ou plusieurs blocs d'arborescence
- Un ou plusieurs images « bootloader »

❑ Installation sur matériel;

- Pas de méthode standard,
- Dépend du matériel,
- Buildroot génère des images (SD card, USB Key)



100

Système Embarqué

(Linux Embarqué)



make menuconfig



```
Buildroot 2015.02 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, </> for Help, </> for Search. Legend: [*] feature is selected [ ] feature is
```

```
Target options --->
Build options --->
Toolchain --->
System configuration --->
Kernel --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options --->
```



<Select> < Exit > < Help > < Save > < Load >

Système Embarqué

(Linux Embarqué)

make defconfig



- Buildroot est livré avec un certain nombre de configurations de bases pour diverses plates-formes matérielles :
 - RaspberryPi,
 - BeagleBone Black,
 - CubieBoard,
 - Atmel evaluation boards,
 - Divers i.MX6 boards,
 - ...
- Construit seulement un système minimum :
 - toolchain,
 - bootloader,
 - Kernel
 - root filesystem minimum

Système Embarqué

(Linux Embarqué)

TP 2:

En suivant les étapes citées ou encore à l'aide des tutoriaux:

- Configurer et compiler un nouveau noyau linux, moyennant l'outil Buildroot,*
- Créer un système d'exploitation complet pour l'embarqué sur
 - Carte Raspberry,**
 - ou*
 - Client Cloud Léger***

Système Embarqué

(Linux Embarqué)



➔ BusyBox:



- ❑ BusyBox est un logiciel libre, écrit en langage C,
- ❑ Distribué sous la licence GNU GPL version 2,
- ❑ Grand nombre des commandes standard sous Unix
- ❑ BusyBox est conçu comme un unique fichier exécutable, ce qui le rend très adapté aux distributions Linux utilisées sur les systèmes embarqués.



104

Système Embarqué

(Linux Embarqué)



➔ BusyBox:



❑ Implantation sur:

➤ Des points d'accès, des routeurs, des téléphones IP,



➤ Générations de robots (AR-Drone).



➤ Livebox, Freebox, Bbox



➤ Des smartphone et tablettes tactiles,...



Système Embarqué ([Linux Embarqué](#))



BUSYBOX

About

- About BusyBox
- Running in VM
- Screenshot
- Announcements

Documentation

- FAQ
- Command Help

Get BusyBox

- Download Source
- Download Binaries
- License
- Products

Development

- Browse Source
- Source Control
- Mailing Lists
- Bug Tracking
- Use less RAM
- Contributors

The Software Freedom Conservancy acts as the GPL enforcement agent for various BusyBox copyright holders. If you wish to report a GPL violation of BusyBox, please write to gpl@busybox.net.

Lic without a vendor

I want to thank the following companies which are providing support for the BusyBox project:

- Analog Devices, Inc. provided a Blackfin development board free of charge. Blackfin is a NMMU processor, and its availability for testing is invaluable. If you are an embedded device developer, please note that Analog Devices has an entire Linux distribution available for download for board. Visit <http://blackfin.uclinux.org> for more information.

9 September 2018 – BusyBox 1.29.3 (stable)

BusyBox_1.29.3_iptl

Bug fix release. 1.29.3 fixes a fix in libbb for armelc_digels()

31 July 2018 – BusyBox 1.29.2 (stable)

BusyBox_1.29.2_iptl

Bug fix release. 1.29.2 has fixes for libbb (compat fixes, allow 2TB+ sizes), gzip (FEATURE_GZIP_LEVELS was producing badly-compressed .gz) files

15 July 2018 – BusyBox 1.29.1 (stable)

BusyBox_1.29.1_iptl

Activate Windows
See Microsoft's site for details

Références

<https://www.busybox.net/>

https://www.busybox.net/live_bbox/live_bbox.html

<https://www.busybox.net/downloads/BusyBox.html>

<https://fr.wikipedia.org/wiki/BusyBox>

<https://www.youtube.com/watch?v=KQ48WBtH8>

Système Embarqué ([Linux Embarqué](#))



→OpenEmbedded:

❑Framework pour Linux embarqué.

❑OpenEmbedded offre un environnement cross-compile le meilleur de sa catégorie.

❑Il permet aux développeurs de créer une distribution Linux complète pour les systèmes embarqués.

❑Avantages d'OpenEmbedded,:
 •Adopté comme système de construction du projet Yocto en mars 2011.
 •Prise en charge de nombreuses architectures matérielles,
 •Versions multiples pour ces architectures,
 •Facile à personnaliser,
 •S'exécute sur toute distribution Linux...

Références

http://www.openembedded.org/wiki/Main_Page

<http://www.kernel-labs.org/files/openembedded-guide/openembedded-guide.html>

<http://docs.openembedded.org/usermanual/html/>

<https://www.youtube.com/watch?v=mOZSpYzpfk&list=PLingpsXjKIRcCVzcVI55IQrcZ8fLMKkN>



TinyOS



108

Système Embarqué [\(TinyOS\)](#)



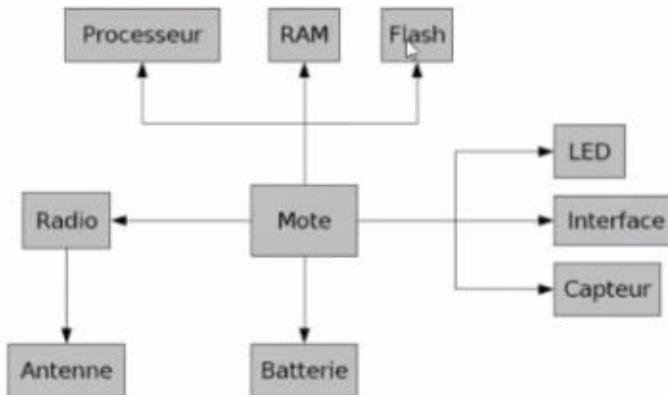
- ❖ **TinyOS** est un système Embarqué, conçu pour des réseaux de capteur sans fil.
- ❖ Il respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place.
- ❖ bibliothèque de composant :
 - protocoles réseaux.
 - pilotes de capteurs
 - outils d'acquisition de données.
- ❖ TinyOS s'appuie sur un fonctionnement événementiel.
- ❖ propose à l'utilisateur une gestion très précise de la consommation du capteur.



109

Système Embarqué

(TinyOS)



Architecture générale des cibles utilisant TinyOS

110

Système Embarqué

(TinyOS)



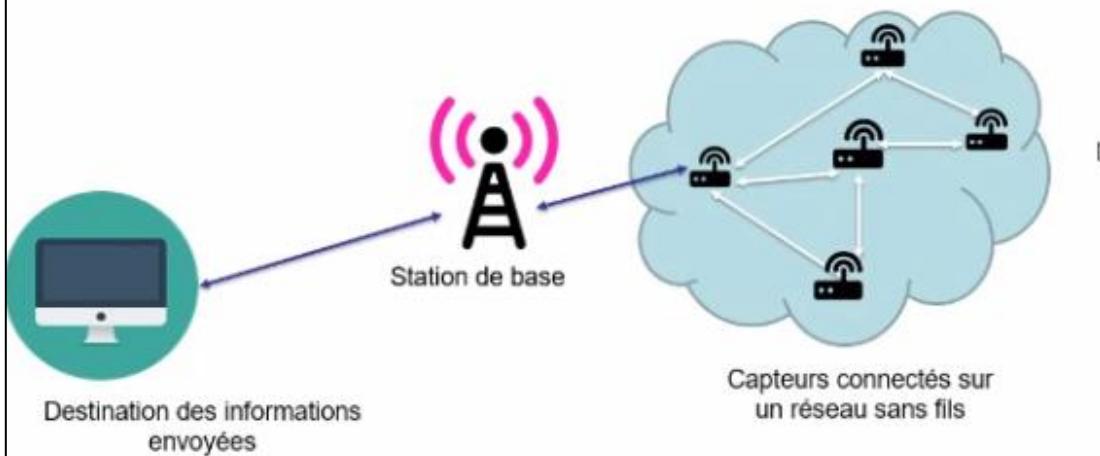
- **Mote, processeur, RAM et Flash** : On appelle généralement Mote la carte physique utilisant TinyOS pour fonctionner, il représente la base du calcul binaire et du stockage.
- **Radio et antenne** : permet la connexion entre l'équipement et la couche physique constitué par les émissions hertziennes.
- **LED, interface, capteur** : on retrouve donc des équipements bardés de différents types de détecteurs et autres entrées.
- **Batterie** : nécessité d'une alimentation autonome.

111

Système Embarqué

(TinyOS)

- Un capteur est un nœud dans un réseau sans fils des capteurs.



112

Système Embarqué

(TinyOS)

Exemple de capteurs:

- type : Mica2 et MicaDot



- Iris (détection de mouvement)

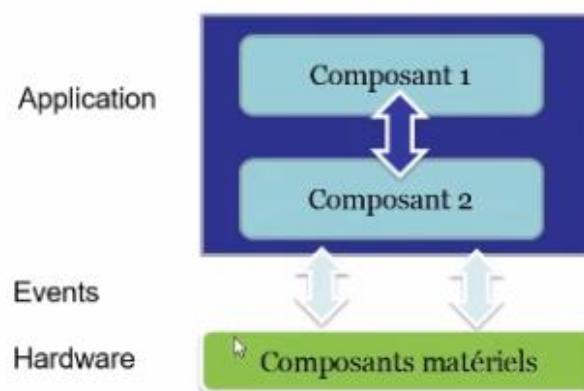


113

<https://www.tinkercad.com/>

Système Embarqué (TinyOS)

- Tinyos appuie sur la langage **NesC** (Network Embedded System C).
- Architecture basé sur les **composants**.
- Une application Tinyos est un ensemble des composants associés dans un but précis.



Systèmes Temps réel

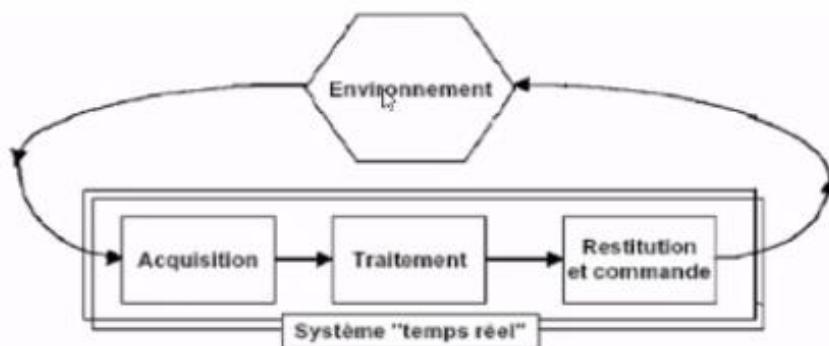
137



Systèmes Temps réel

DEFINITION:

- Application mettant en œuvre un système informatique dont le fonctionnement est assujetti à l'évolution dynamique d'un procédé extérieur qui lui est connecté et dont il doit contrôler le comportement.



138



Systèmes Temps réel

Domaines d'application:

- Suivi de position d'objets par satellites,
- Pilotage de chaînes de montage dans des usines automatisée
- Contrôle de procédés industriels (mécanique, chimie,...)
- Applications embarquées (assistance au pilotage, contrôle de trajectoire, ...)
- Applications mobiles (audio, vidéo, ...)
- Services Web...

139



Systèmes Temps réel

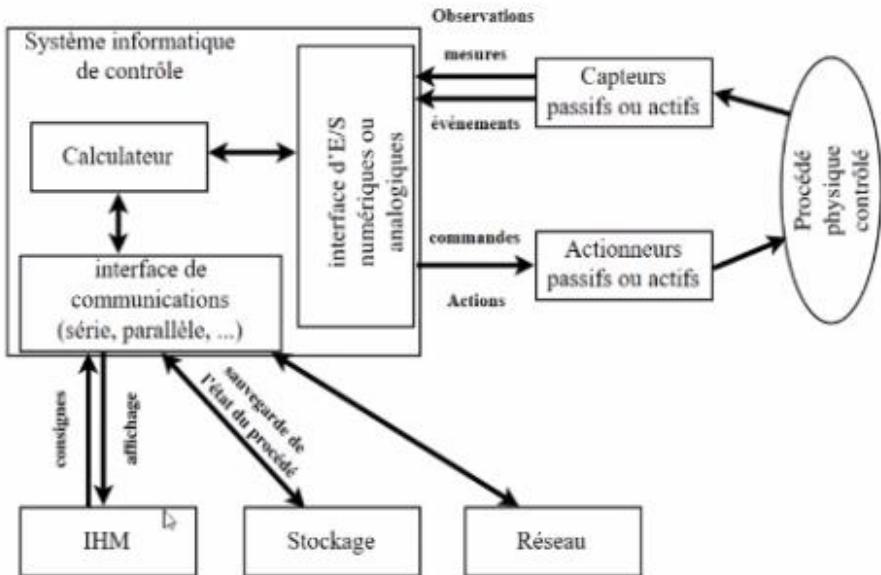
Contraintes Temps réel: ordre de Grandeur

- Mesures scientifiques : 10ns - 1ms,
- Systèmes radar : 1μs - 1ms
- Systèmes vocaux : 10μs - 10ms
- Robotique : 1ms - 10 ms
- Contrôle de stockage : 1s - 1mn
- Contrôle de fabrication : 1mn - 1h
- Contrôle de réactions chimiques : 1h et plus
- Mission spatiale : mois ou années

140

Systèmes Temps réel

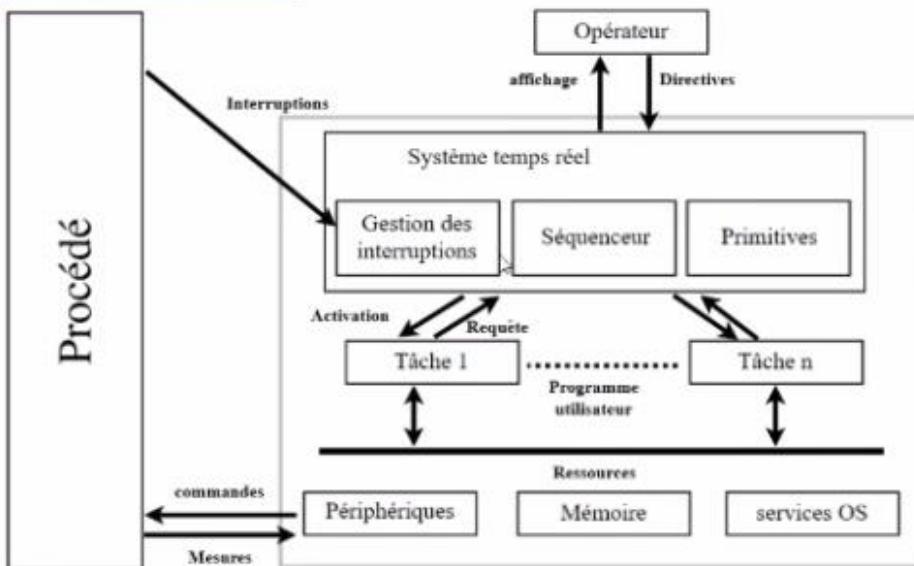
Architecture d'un STR:



141

Systèmes Temps réel

Système Multi-tâches:

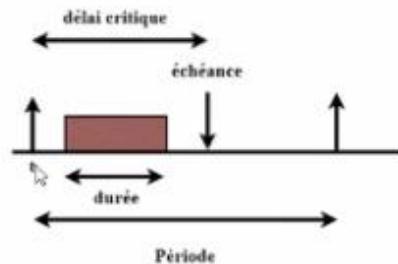


142

Systèmes Temps réel

Système Multi-tâches: (Tâches)

- Indépendantes ou coopératives (processus/threads)
- Caractérisées par leur contraintes temporelles :
 - *date de réveil*
 - *durée d'exécution*
 - *délai critique*



143

Systèmes Temps réel

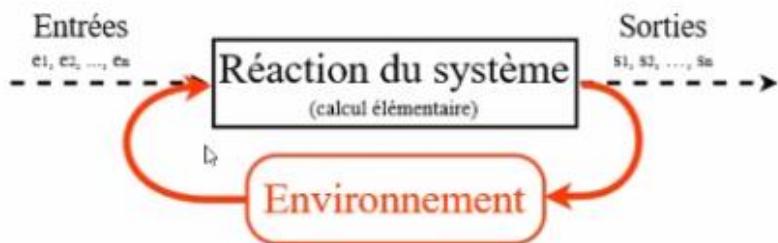
Système Réactifs:

- *Systèmes transformationnels* : processus acceptant des données au démarrage et produisant des résultats à la fin.
 - **EXP:** Une procédure, une fonction, une commande (cp, LaTeX, ...)
- *Systèmes interactifs* : processus interagissant de façon répétitive en répondant à des solicitations lorsqu'ils sont prêts à les traiter. La solicitation fixe les entrées à traiter et les résultats sont produits après traitements.
 - **EXP:** Systèmes d'exploitation, un shell UNIX, GUI de station de travail, Jeux, ...
- *Systèmes réactifs* : processus toujours prêts à réagir à de nouvelles entrées (interaction constante avec l'environnement).
 - **EXP:** Surveillance, Traitement du signal, Protocoles de communication, IHM embarquée

144

Systèmes Temps réel

Système Réactifs:



- La réaction du système est caractérisée par la fonction entre les entrées
- $\{e_1, \dots, e_n\}$ et les sorties $\{s_1, \dots, s_n\}$