Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Systèmes Temps Réel

CH1: Introduction aux Systèmes Temps Réel

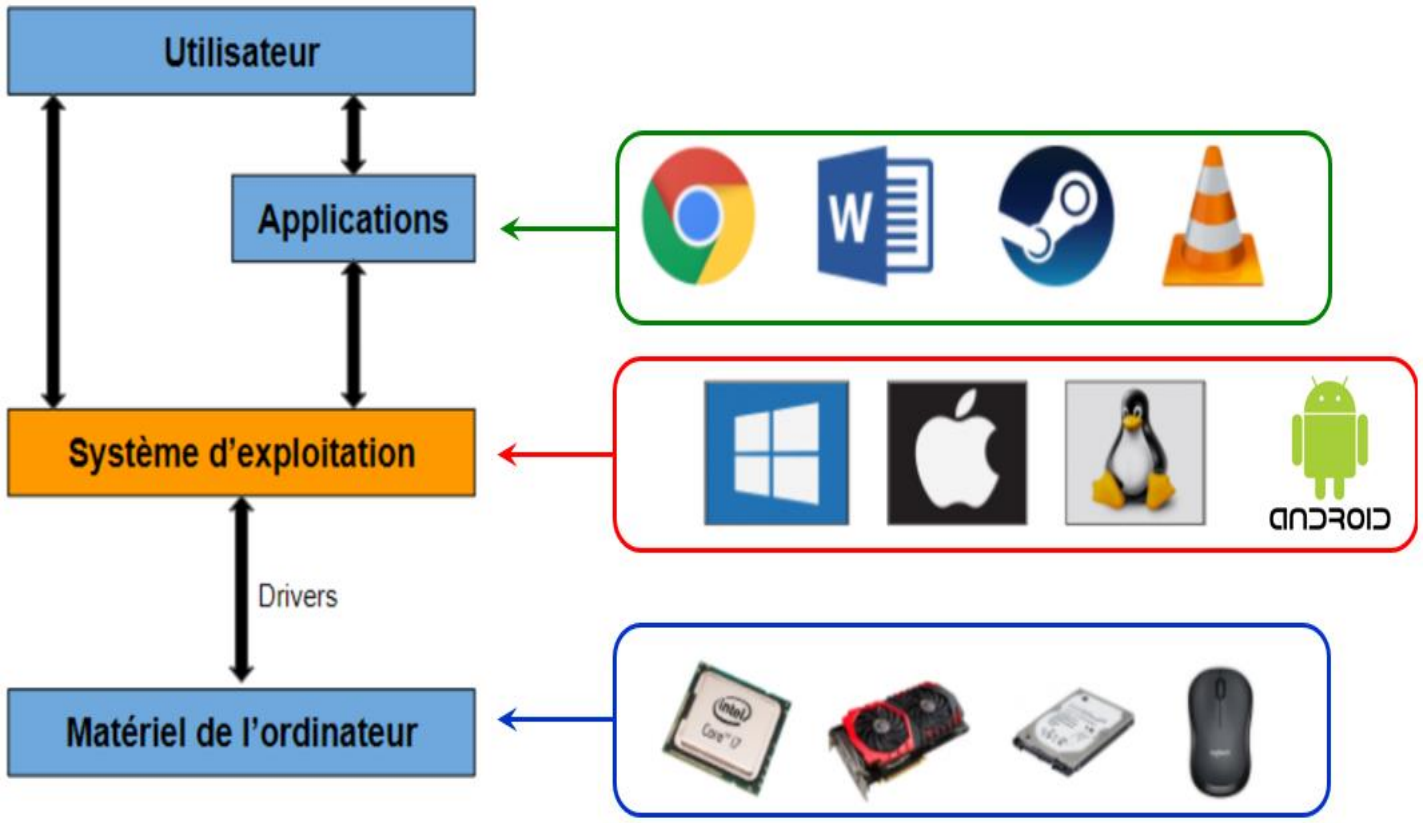
Par:

Hassan EL FADIL

elfadil.h@gmail.com

1. Rôle d'un système d'exploitation

- Un système d'exploitation (OS pour Operating System en anglais) est un logiciel qui agit comme une interface entre les utilisateurs, les applications et le matériel d'un ordinateur ou d'un dispositif informatique.
- Son rôle principal est de gérer les ressources matérielles et logicielles de manière efficace pour permettre le bon fonctionnement du système informatique.



2. Définition d'un système d'exploitation temps réel (RTOS)

- Système d'exploitation temps-réel (SETR): Real Time Operating System (RTOS)
- C'est une classe de systèmes d'exploitation conçus pour répondre aux contraintes temporelles strictes des systèmes temps réel.
- Contrairement aux systèmes d'exploitation traditionnels (GPOS: General Purpose Operating System) qui mettent l'accent sur la convivialité et la gestion efficace des ressources, les RTOS sont optimisés pour garantir des réponses rapides et prédictibles aux événements.
- Les délais peuvent être aussi courts que quelques microsecondes ou millisecondes, selon les exigences du système.

3. La contrainte temps dans un RTOS

Le comportement d'un RTOS dépend essentiellement :

De l'exactitude des traitements effectués.

Exemple suivi exact de la trajectoire d'un missile

Du temps requis pour produire le résultat.

*Exemple: temps maximal de déclenchement de l'airbag de **30 à 150ms** (en fonction du modèle de la voiture)*

Dans un RTOS, un résultat juste mais hors délai est un résultat erroné!

4. GPOS vs RTOS

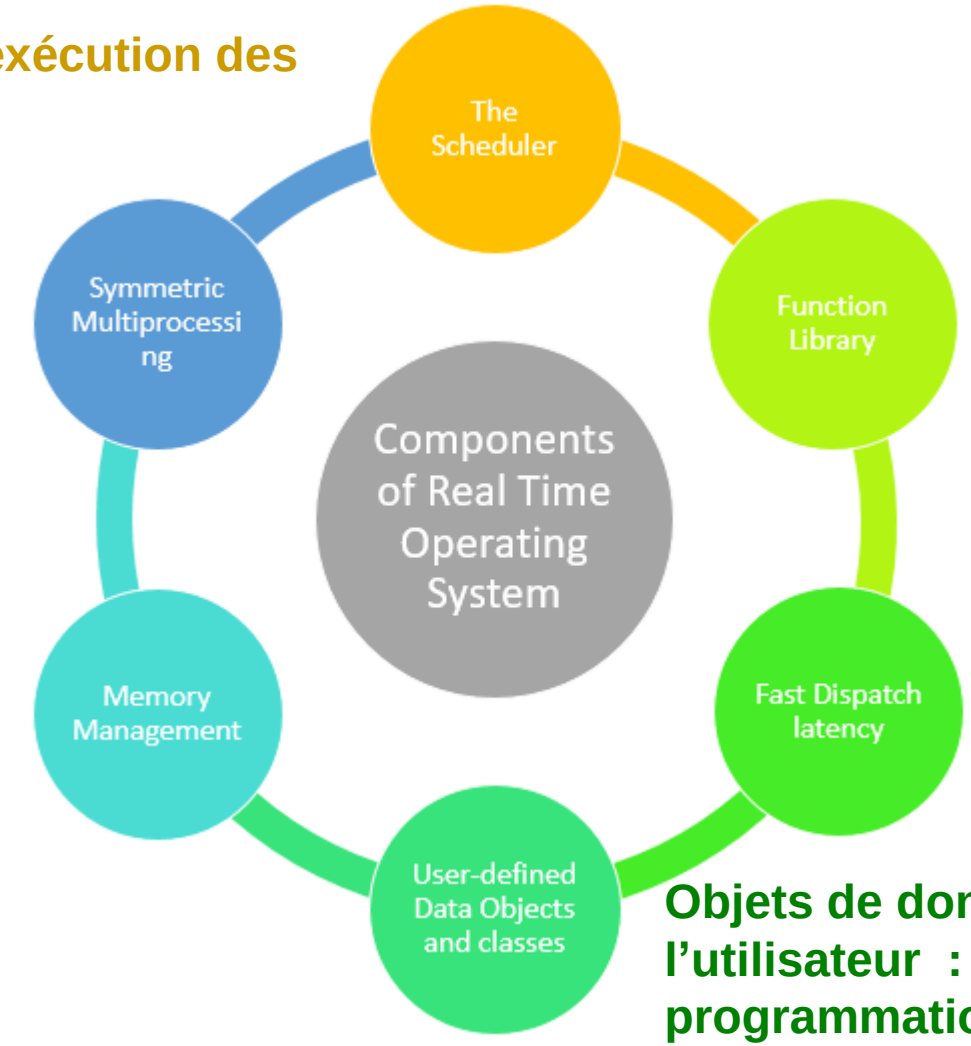
Aspect	GPOS (Système d'exploitation général)	RTOS (Système d'exploitation temps réel)
Priorité temporelle	Moins critique, réponses dans des délais variables	Critique, réponses dans des délais stricts
Planification des tâches	Basée sur l'efficacité et l'optimisation des ressources. Non prédictible (on ne sait pas quelle tâche sera exécutée et pour combien de temps)	Basée sur les priorités et les échéances temporelles (préemptif)
Gestion des interruptions	Généralement gérées avec un certain délai	Gérées avec une latence minimale
Latence	Peut avoir des temps de réponse variables	Temps de réponse prédictibles et courts
Taille des noyaux	Souvent plus grands, offrent de nombreuses fonctionnalités	Plus compacts, optimisés pour la rapidité
Multitâche/multi-utilisateur	Multitâche/multi-utilisateur	Multitâche mais non multi-utilisateur
Interface utilisateur	Présente une interface utilisateur (graphique GUI ou ligne de commande CLI)	Spécifiquement conçu et optimisé pour les systèmes embarqués
Exemples	Windows, macOS, Linux, Android	FreeRTOS, VxWorks, RTLinux, QNX, Micrium µC/OS

5. Composants du RTOS

L'ordonnanceur: Gestion de l'exécution des tâches par priorité

Traitement Symétrique: Nombre de tâches différentes qui peuvent être gérées par le RTOS afin qu'un traitement parallèle puisse être effectué.

Gestion de la mémoire: Nécessaire dans le système pour allouer de la mémoire à chaque programme



Bibliothèque de fonctions: Interface qui permet de connecter le code du noyau et de l'application

Latence de réponse rapide: temps qu'il faut pour qu'une tâche de haute priorité soit exécutée après avoir été préemptée par le noyau.

Objets de données et classes définie par l'utilisateur : utilisation des langages de programmation comme C ou C++

6. Types des RTOS

RTOS Dur (Hard RTOS)

- Dans un RTOS dur, les contraintes de temps sont strictes et absolues.
- Les tâches critiques doivent être exécutées dans des délais spécifiques, sinon le système peut échouer ou produire des résultats incorrects: donc risques.
- On parle d'un RTOS **déterministe**
- Exemples: détection de missile, airbag, ABS, pilotage automatique des avions, un défibrillateur, etc.

RTOS Ferme (Firm RTOS)

- Les RTOS fermes ont également des contraintes de temps, mais ils sont plus flexibles que les RTOS durs. Ils peuvent tolérer des dépassements de délai occasionnels sans compromettre la fonctionnalité du système.
- Les RTOS fermes sont utilisés dans des applications où les contraintes de temps sont importantes mais où des retards occasionnels peuvent être tolérés sans compromettre la mission
- Exemples: systèmes de télécommunications, contrôle industriel, etc.

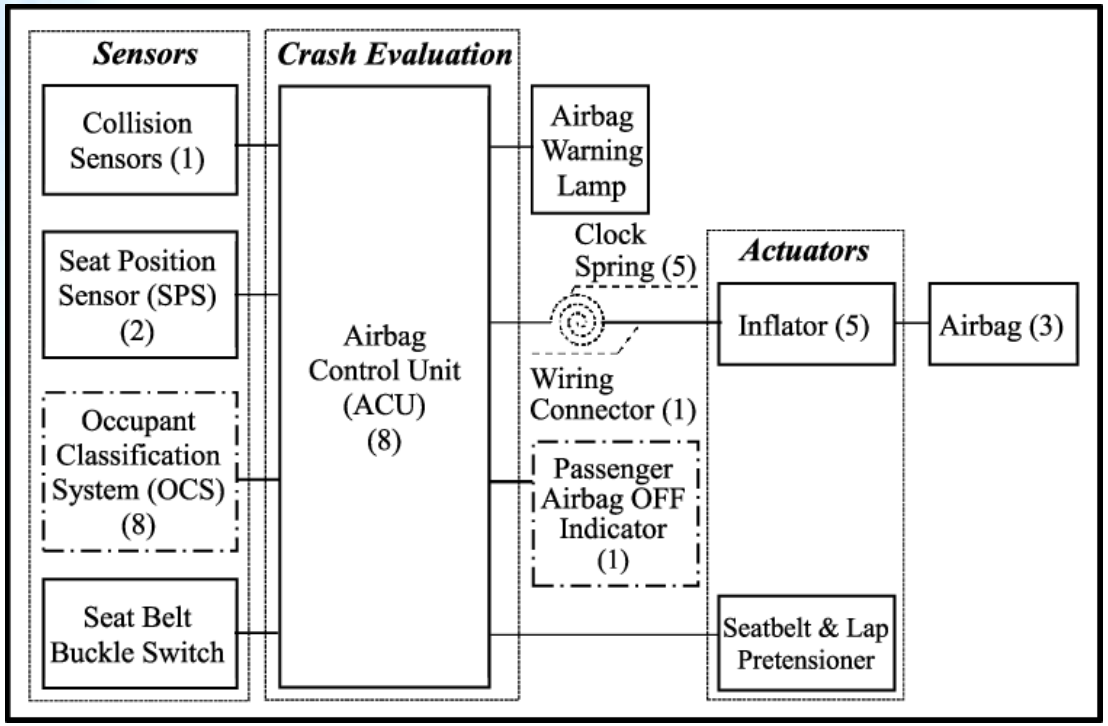
RTOS Souple (Soft RTOS)

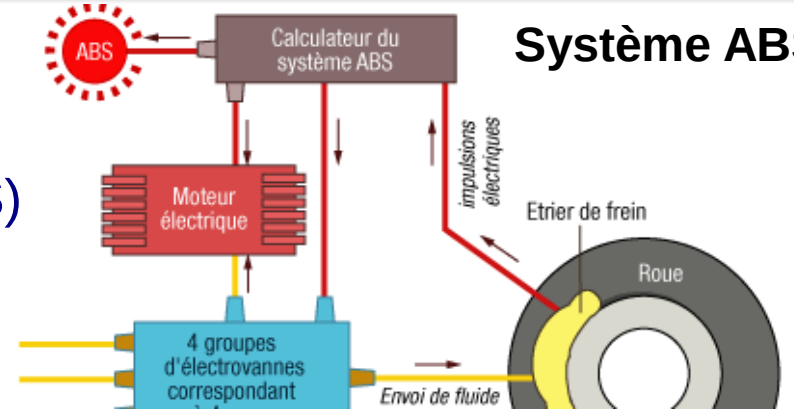
- Les RTOS souples ont des contraintes de temps moins strictes et sont plus tolérants aux retards.
- Ils visent à fournir des performances temps réel lorsque cela est possible, mais peuvent accepter des retards sans causer de préjudice grave au système.
- Les RTOS souples sont utilisés dans des applications où la réactivité en temps réel est importante mais où des retards peuvent être tolérés sans compromettre le fonctionnement global,
- Exemples: les systèmes de gestion de données, jeux vidéo, distributeurs, etc.

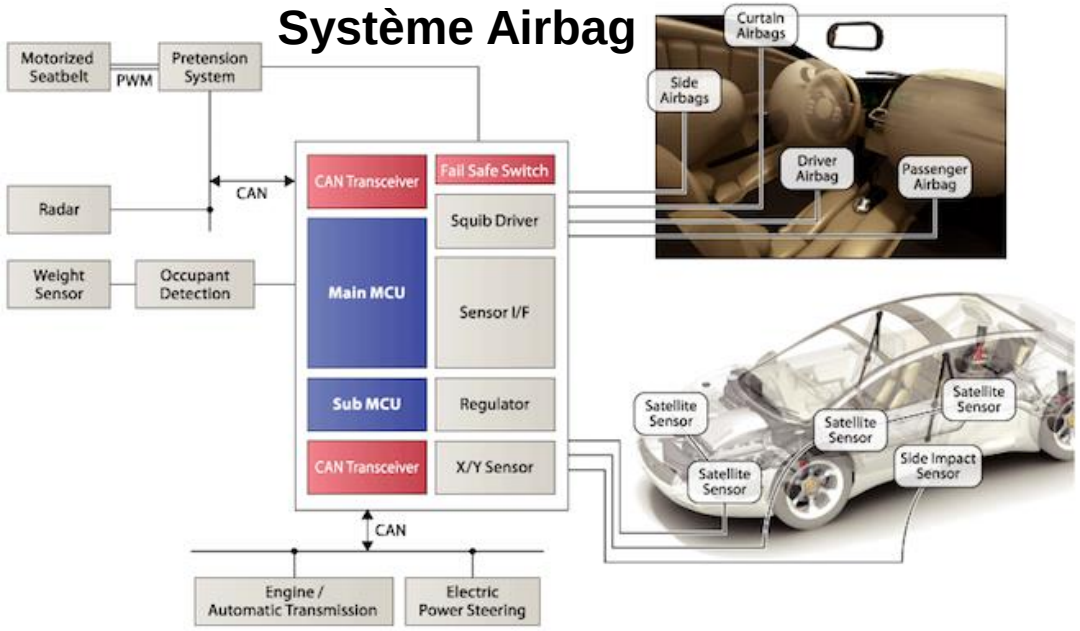
7. Exemples d'applications des RTOS

Automobile

- contrôle du moteur
- gestion des airbags
- système de freinage antiblocage (ABS)
- système de navigation, etc.

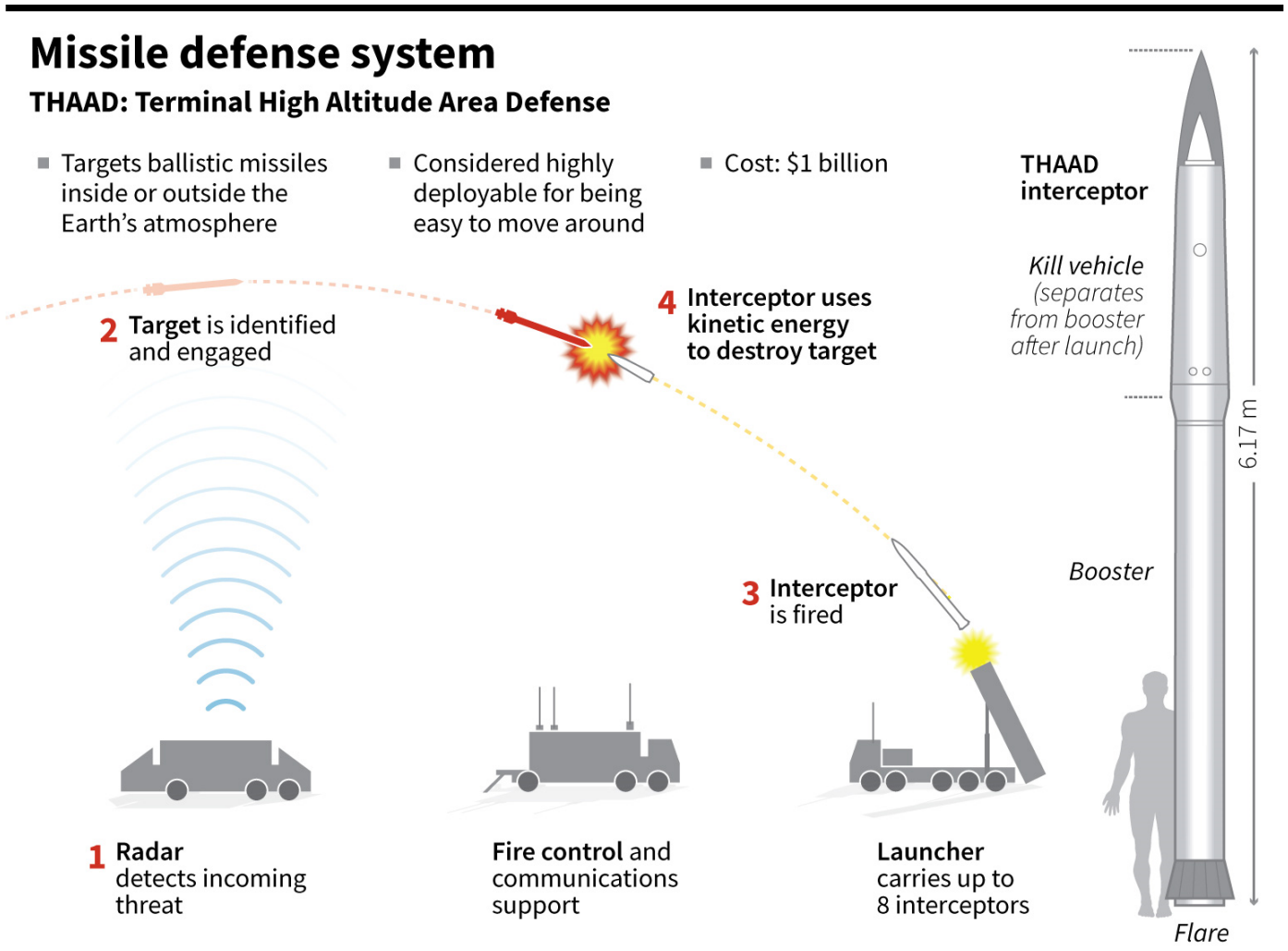






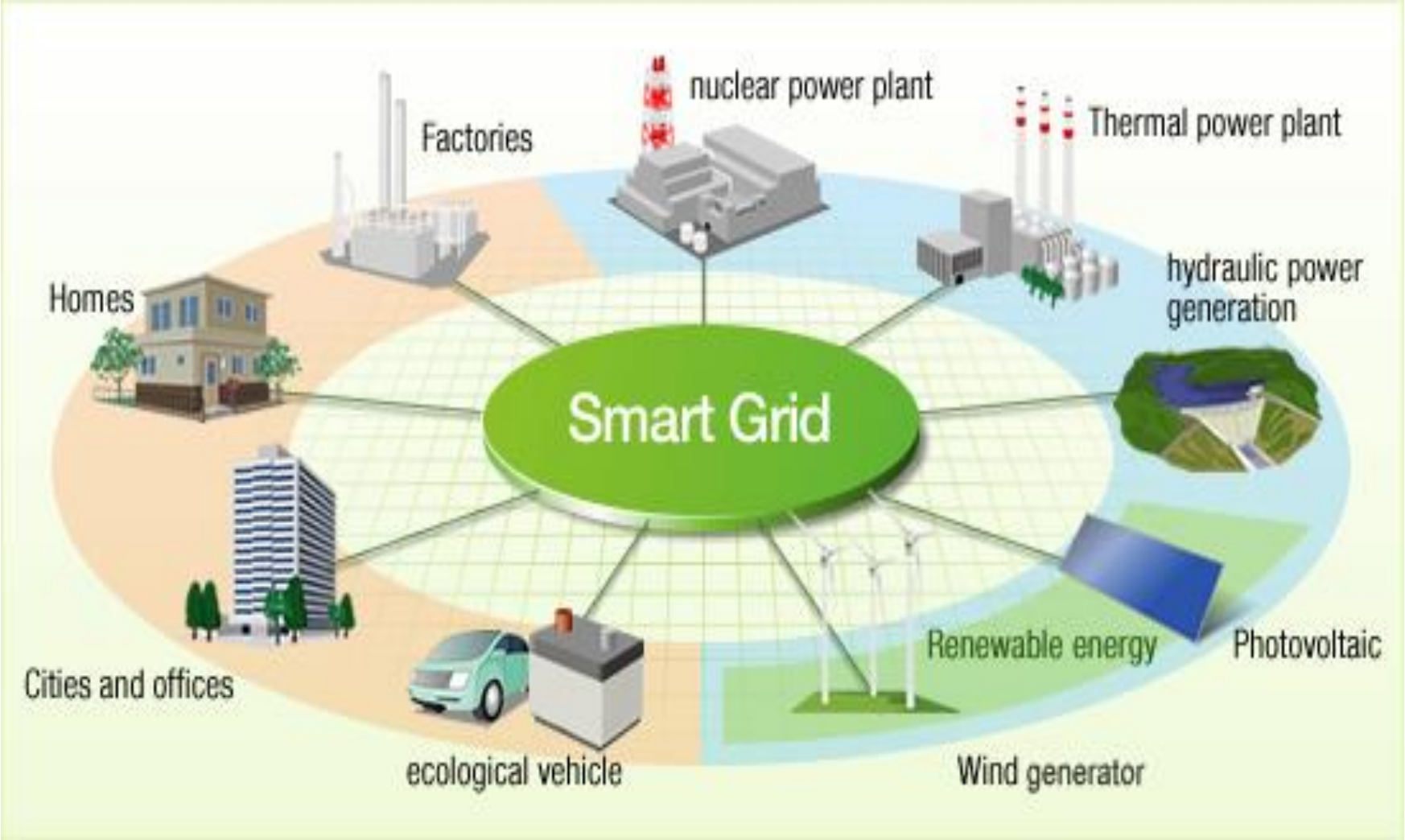
Aérospatiale et défense

- les drones
- les satellites
- les équipements militaires, etc.



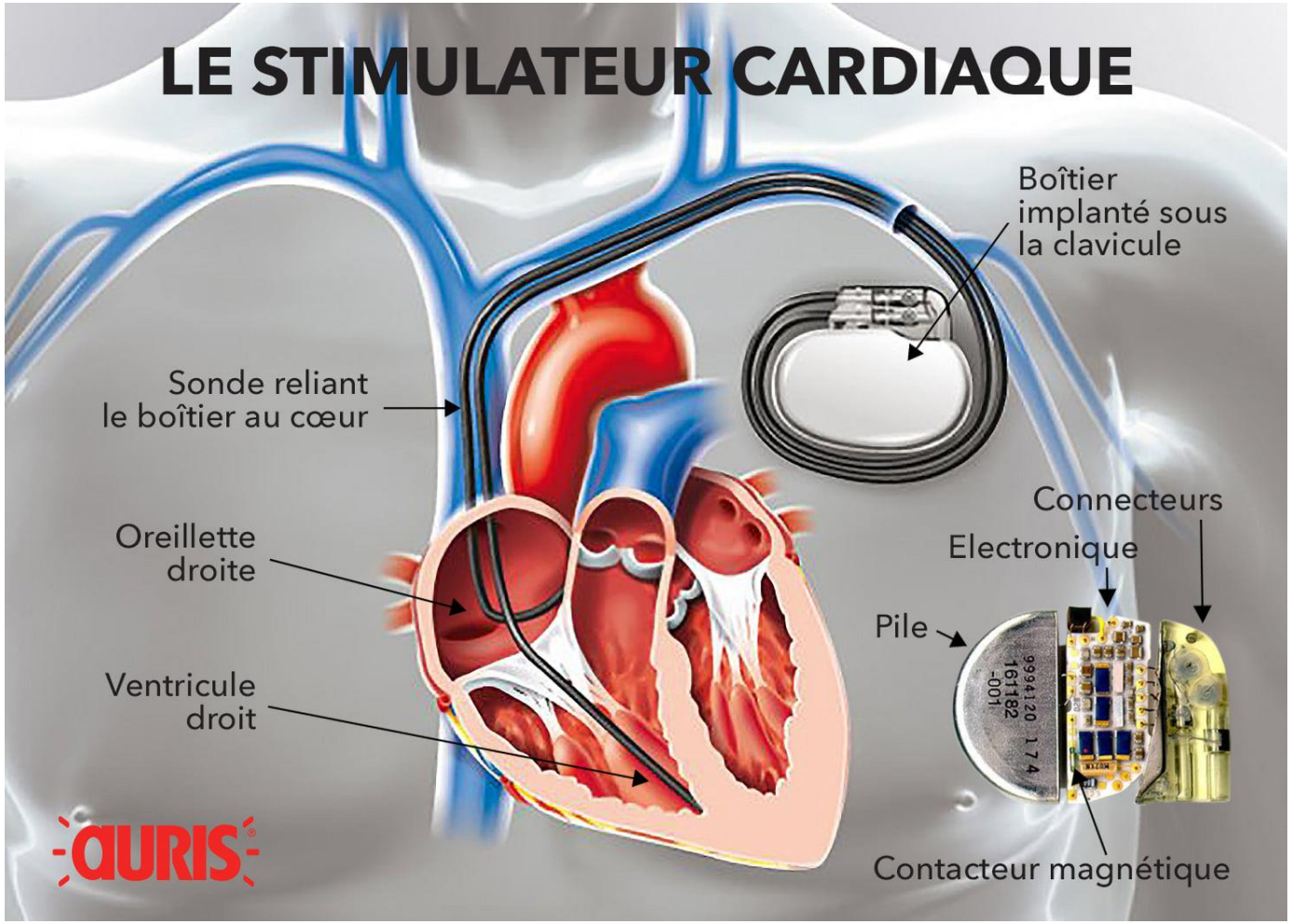
Énergie

- Les systèmes de gestion de l'énergie,
- les réseaux intelligents,
- les compteurs intelligents, etc.



Médecine

- Appareils d'imagerie médicale,
- les pompes à perfusion
- les robots chirurgicaux
- Stimulateur cardiaque (pacemaker)



Télécommunications, IoT

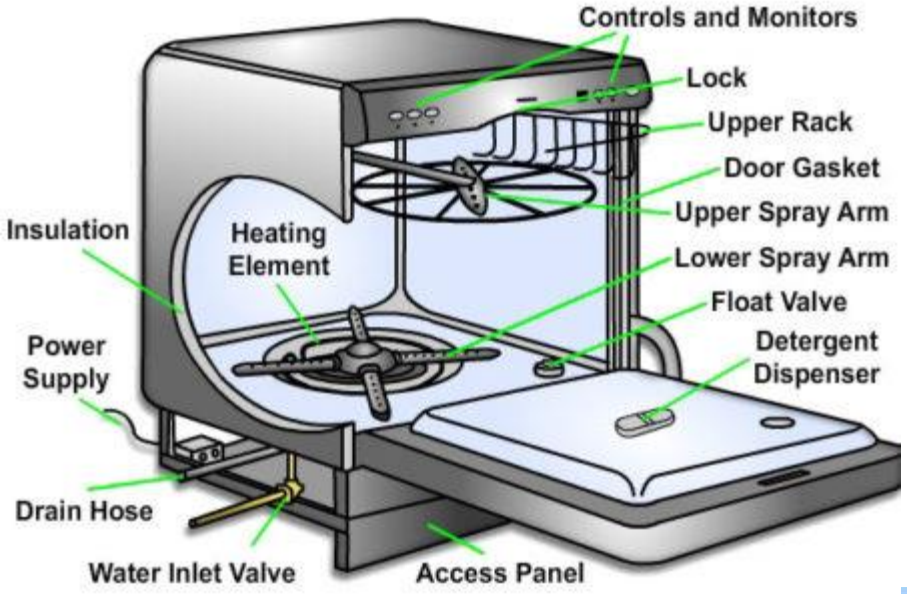
- réseaux de téléphonie mobile,
- les routeurs et les switches
- Les modems
- IoT, etc.

Industrie

- Les machines industrielles automatisées
- les robots de fabrication,
- les systèmes de contrôle de processus,
- les équipements de contrôle qualité

Electroménager

- Lave-vaisselle et lave-linge
- Réfrigérateurs et congélateurs
- Cafetières et bouilloires électriques
- Aspirateurs robots
- Micro-ondes



8. Quelques exemples des RTOS

FreeRTOS : FreeRTOS est l'un des RTOS les plus populaires et largement utilisés.

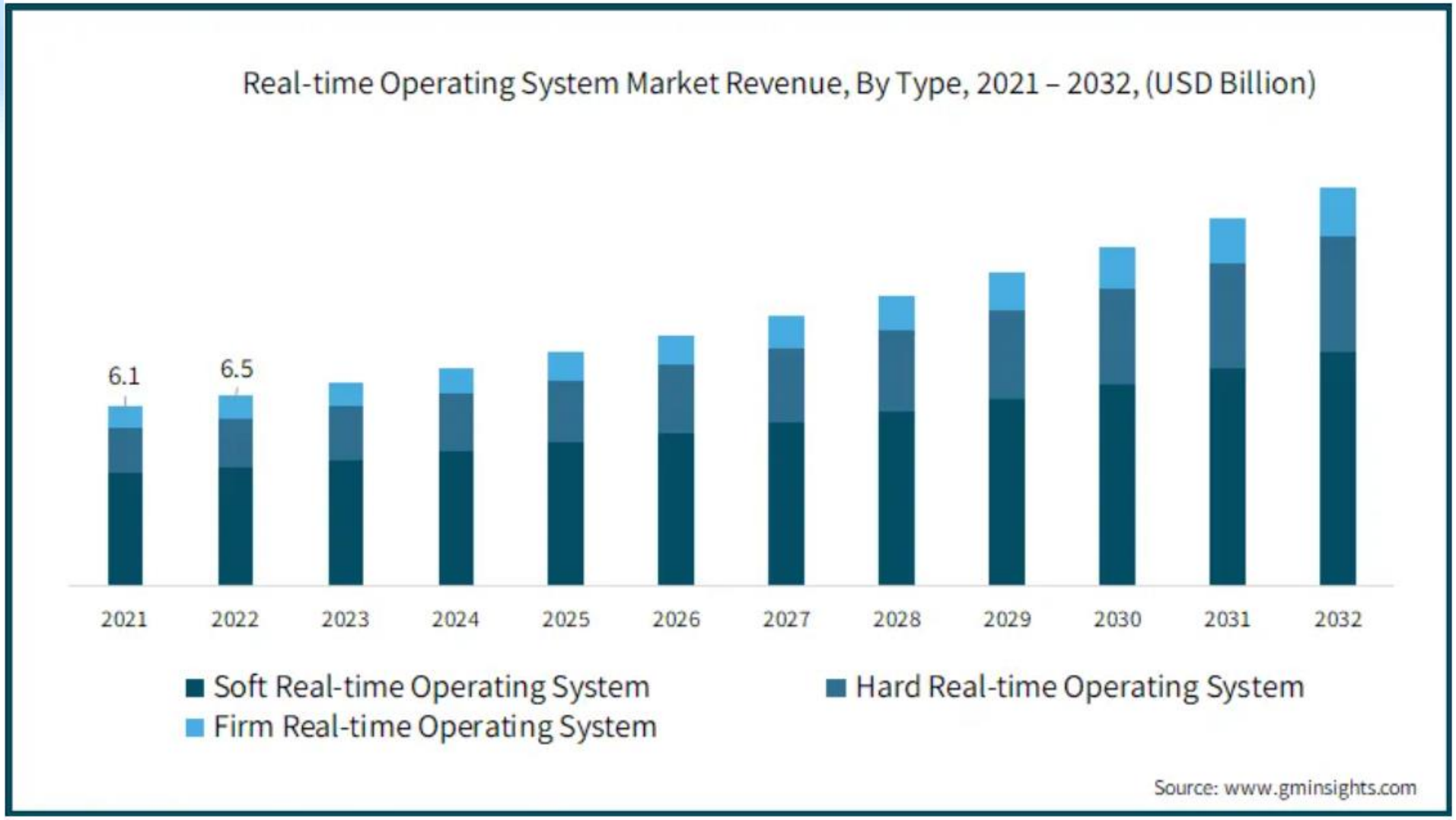
VxWorks : VxWorks est un RTOS commercial largement utilisé dans des domaines tels que l'aérospatiale, la défense, les télécommunications, l'automobile, l'électronique grand public et les systèmes critiques nécessitant une grande fiabilité et une faible latence

QNX : QNX est un autre RTOS commercial réputé, principalement utilisé dans l'automobile, l'aérospatiale, les télécommunications, les appareils médicaux

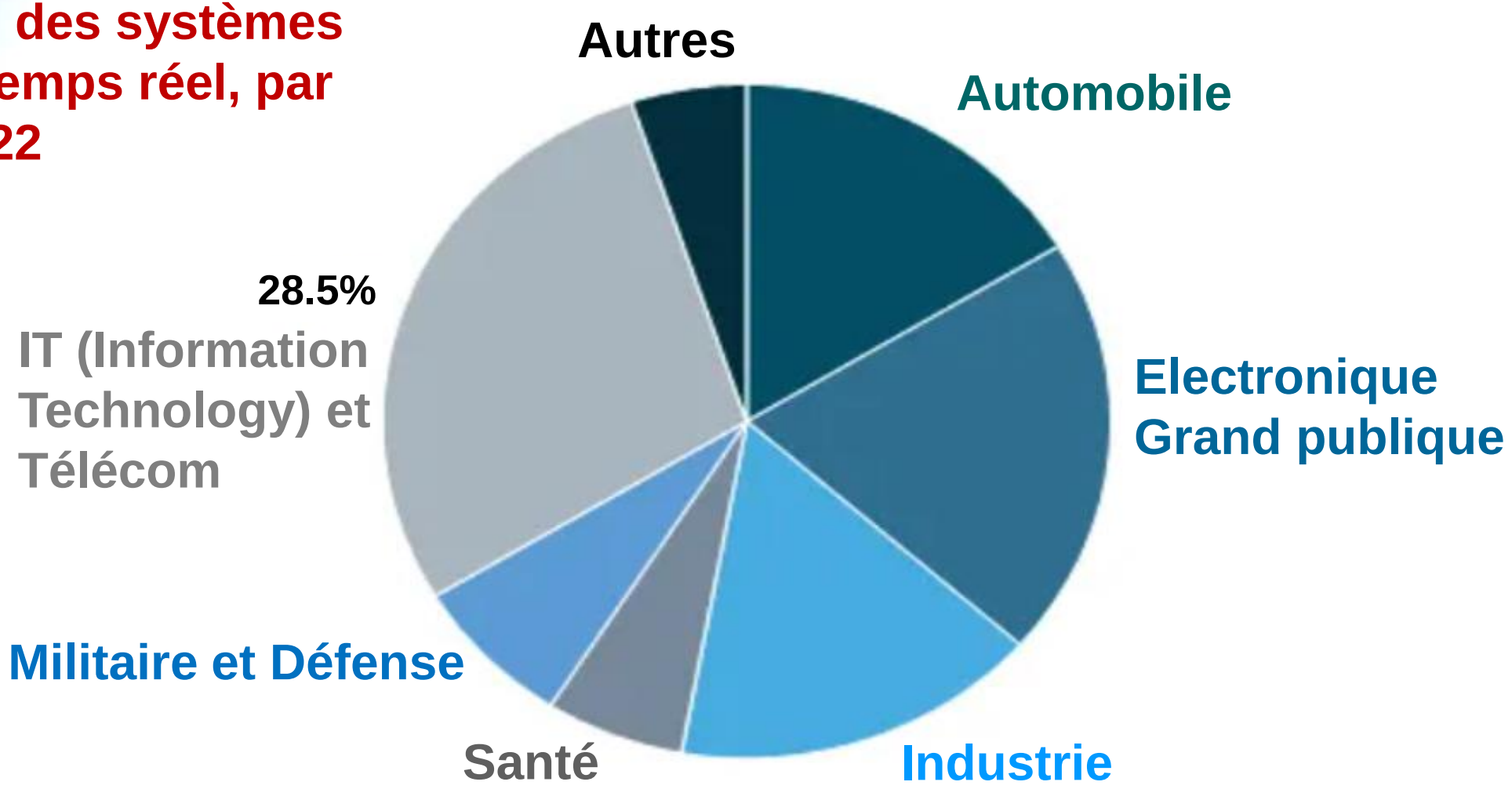
INTEGRITY RTOS: utilisé dans des applications critiques telles que l'aérospatiale, la défense, l'automobile, l'électronique grand public

RTLinux : RTLinux est un RTOS open source basé sur Linux

9. Marché des RTOS



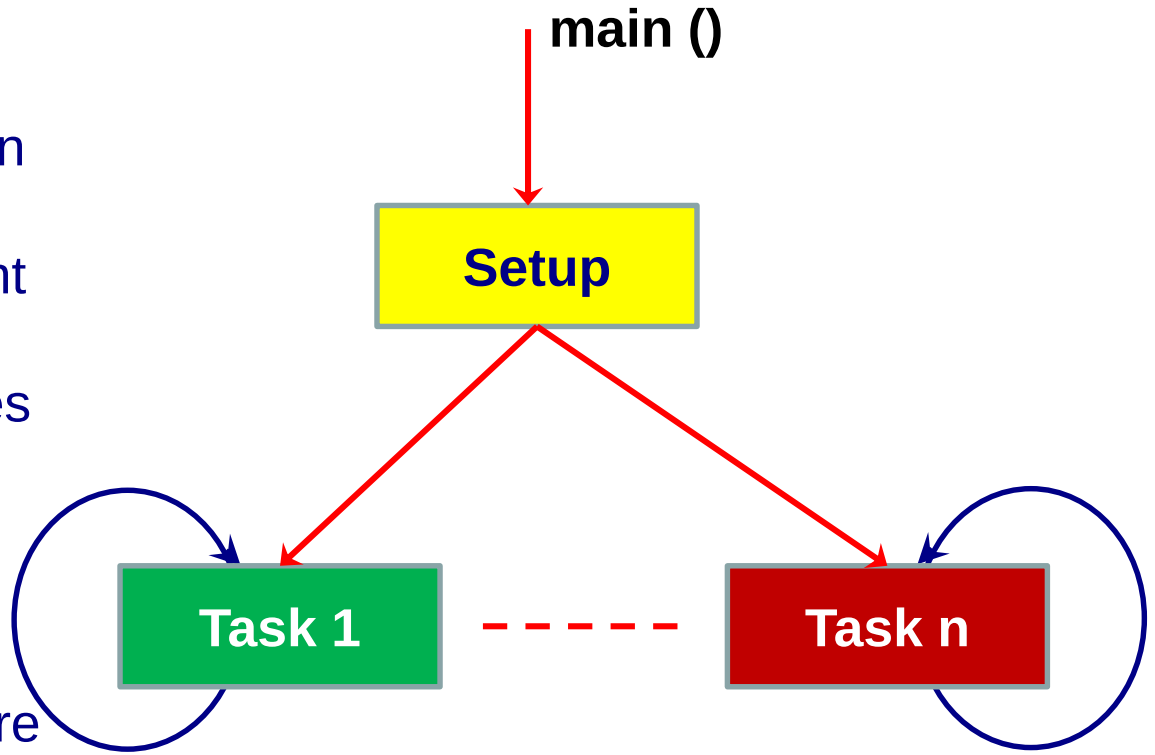
Part de marché des systèmes d'exploitation temps réel, par application, 2022



10. Terminologie RTOS

10.1. Tâche (Task)

- Une tâche est une unité de travail ou d'exécution d'un programme.
- C'est une fonction avec une **boucle infinie** contenant le code d'exécution de la tâche.
- On attribue à chaque tâche **une priorité**. Les priorités permettant au RTOS de hiérarchiser l'exécution des tâches en fonction de leur importance relative.
- Chaque tâche a son propre **contexte d'exécution**, comprenant notamment son état, ses variables locales et les informations nécessaires pour reprendre son exécution après une interruption ou un changement de tâche.
- Chaque tache possède **sa propre zone mémoire** pile (Stack) pour sauvegarder ses données
- Task = Thread (unité d'exécution)

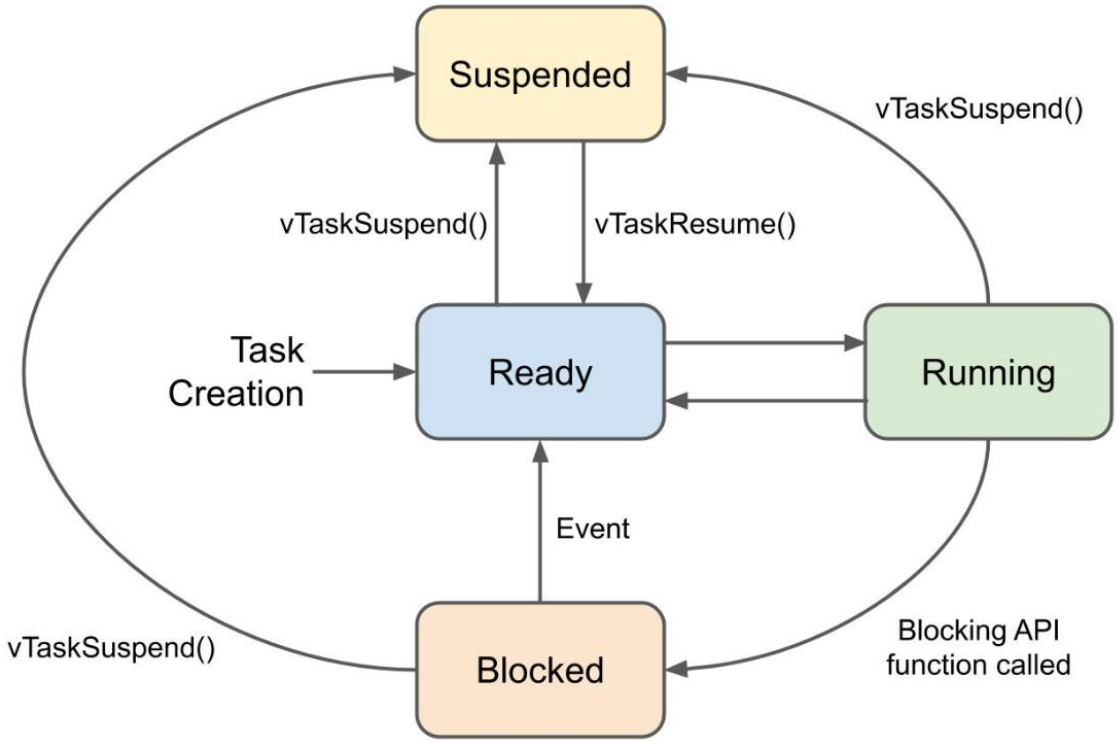


Exemple: Système de surveillance électrique domestique:

Tâche 1: connexion au Wifi
Tâche 2: mesure de l'électricité consommée
Tâche 3: affichage LCD
Tache 4: mesure de la température
Tâche 5: mesure de l'humidité
Tâche 6: envoi des données au cloud
…

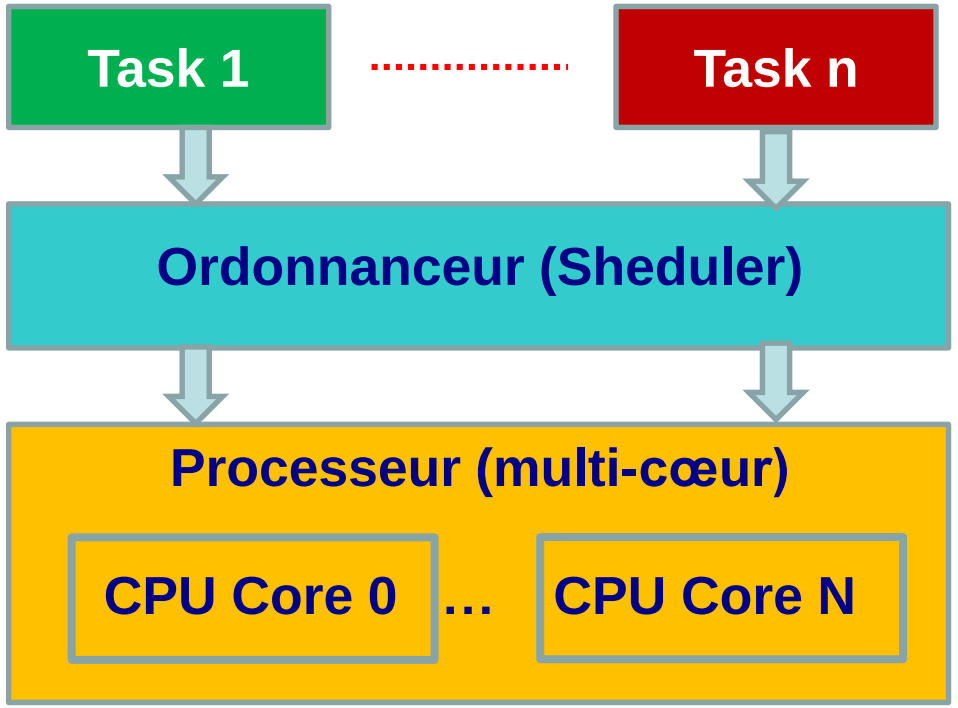
10.2. Etats d'une tâche:

- **Prête (Ready) :** La tâche est prête à s'exécuter mais attend que l'ordonnanceur lui alloue du temps CPU.
- **En cours d'exécution (Running) :** La tâche s'exécute actuellement sur le CPU.
- **Bloquée (ou en attente) (Blocked/Waiting) :** La tâche ne peut pas s'exécuter car elle attend un événement ou une ressource.
- **Suspendue (Suspended) :** La tâche a été temporairement suspendue, soit par elle-même, soit par une autre tâche ou un événement système.
- **Terminée (ou Supprimée) (Terminated/Deleted) :** La tâche a terminé son exécution ou a été explicitement supprimée par le système.



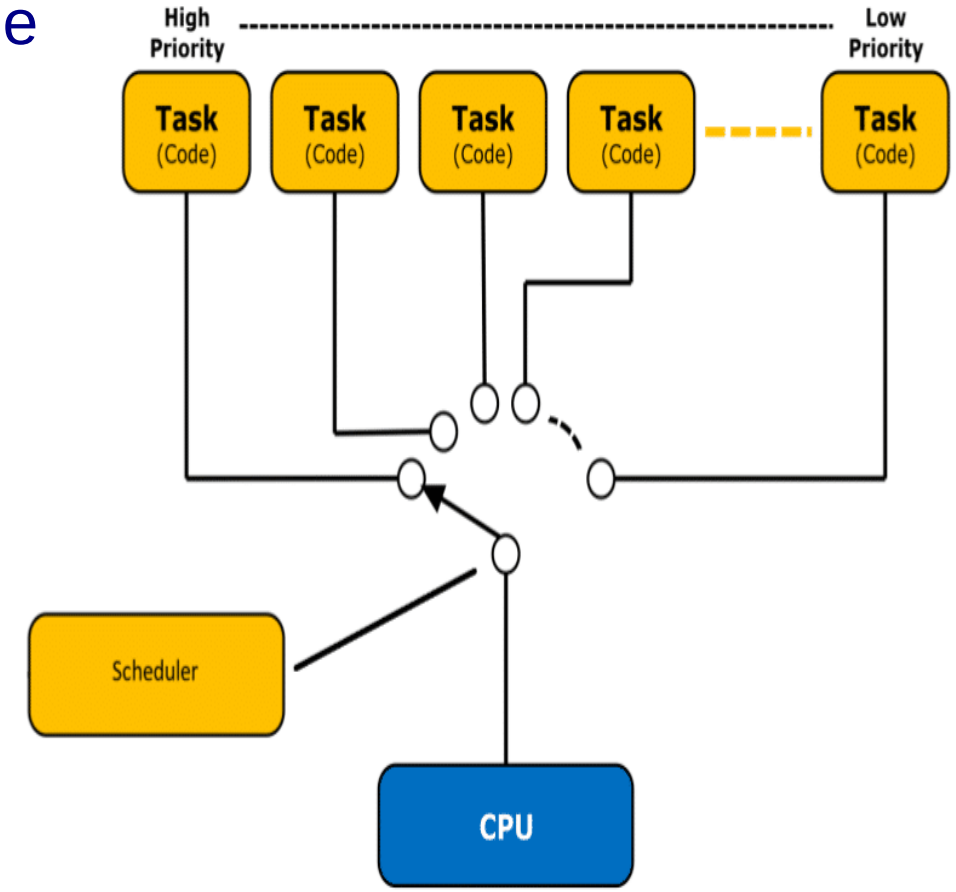
10.3. Multitâche et parallélisme (Multitasking and parallelism)

- **Multitâche :** fait référence à la capacité du système à exécuter plusieurs tâches de manière concurrente. Le multitâche permet d'exploiter efficacement les ressources matérielles en évitant les attentes inutiles et en améliorant la réactivité du système.
- **Parallélisme :** Le parallélisme se réfère à la capacité d'exécuter plusieurs tâches simultanément sur des processeurs multi-cœurs. Dans un RTOS, le parallélisme peut être réalisé en attribuant différentes tâches à différents cœurs de processeur.



10.4. Ordonnanceur (Scheduler):

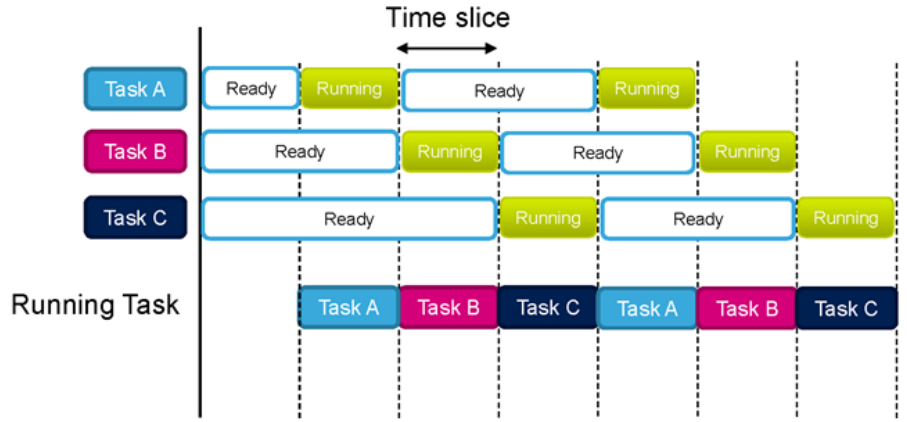
- L'ordonnanceur (scheduler) est responsable de la gestion des tâches et de l'allocation des ressources CPU en fonction des priorités et des contraintes temporelles spécifiques des applications temps réel.
- Il gère les interruptions.
- Un bon ordonnanceur vise à optimiser les performances du système en minimisant les temps d'attente, en maximisant l'utilisation du CPU, et en respectant les contraintes temporelles des applications temps réel.



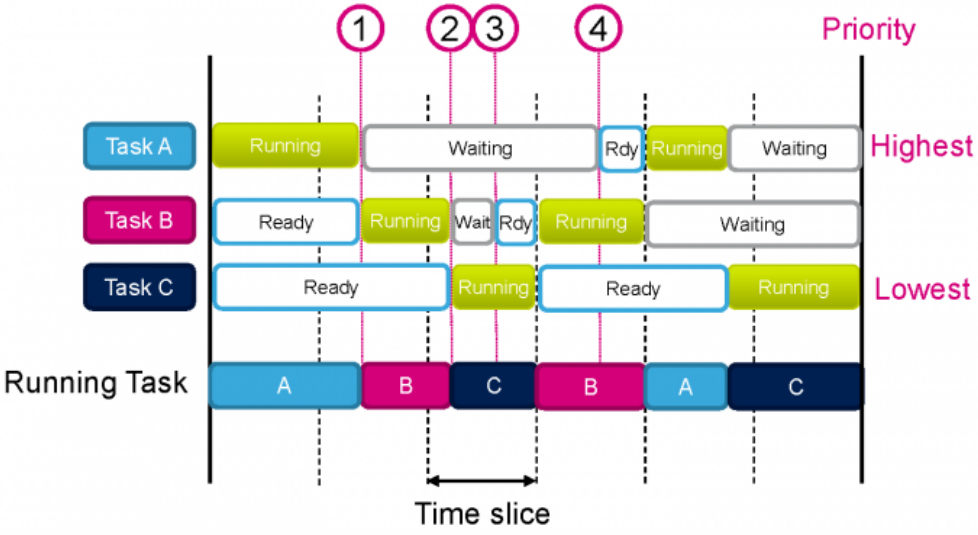
10.5. Types d'ordonnanceurs:

- **Coopératif (Cooperative)** : tâche par tâche, chaque tâche s'exécute jusqu'à ce qu'elle se termine ou qu'elle se mette volontairement en attente.
- **Round-robin** : "tourniquet" ou "rotation circulaire" : chaque tâche dispose d'une tranche de temps pour s'exécuter, il n'y a pas de priorité pour l'exécution des tâches.
- **Préemptif / Basé sur les priorités (Pre-emptive/ Priority-based)**: chaque tâche a une priorité, celle ayant la plus grande priorité peut interrompre la tâche en cours d'exécution et prendre sa place dans l'exécution.

Round-robin scheduler

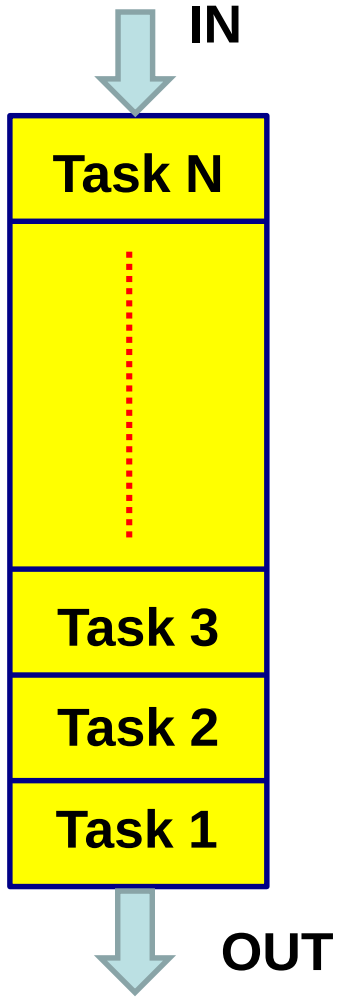


Priority-based scheduler



10.6. File d'attente (Queue):

- Dans les RTOS, une file d'attente (Queue) est une structure de données utilisée pour stocker et gérer des éléments de manière ordonnée selon le principe du premier entré, premier sorti (FIFO - First-In-First-Out).
- Les files sont largement utilisées pour la communication et la synchronisation entre les tâches
- Elles permettent de stocker temporairement des données ou des messages.
- Elles ont généralement une capacité limitée et bien définie.
- Il existe différentes files d'attentes: File d'attente prête (**Ready Queue**), File d'attente bloquée (**Blocked Queue**), File d'attente en cours d'exécution (**Running Queue**).



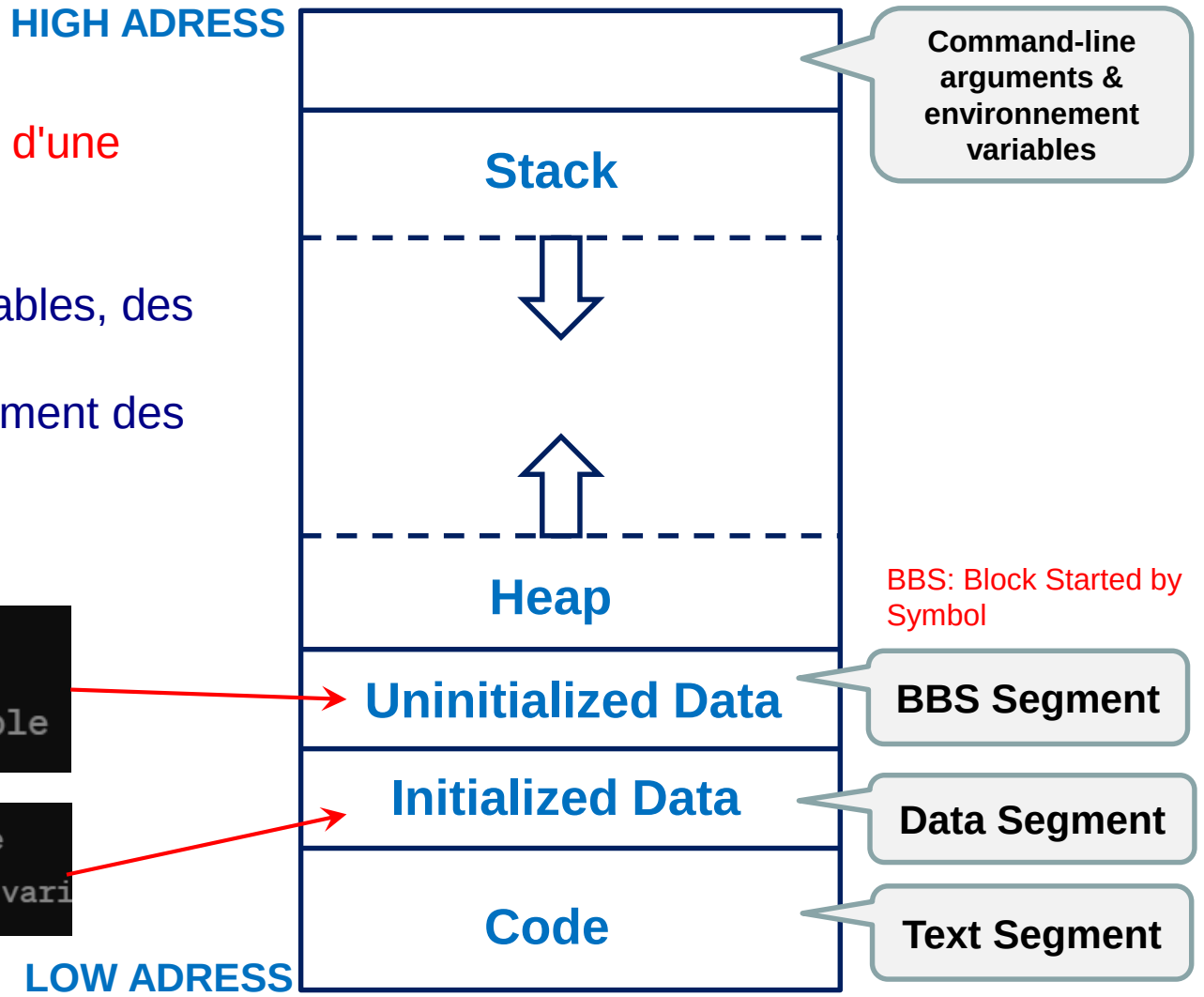
11. Allocation mémoire

11.1. Définition:

- L'**allocation mémoire** est le processus de réservation d'une certaine quantité de mémoire pour l'utilisation par un programme pendant son exécution.
- Cette mémoire peut être utilisée pour stocker des variables, des structures de données, des objets, etc.
- L'allocation mémoire est essentielle pour le fonctionnement des programmes informatiques car elle permet de gérer efficacement l'espace mémoire disponible.

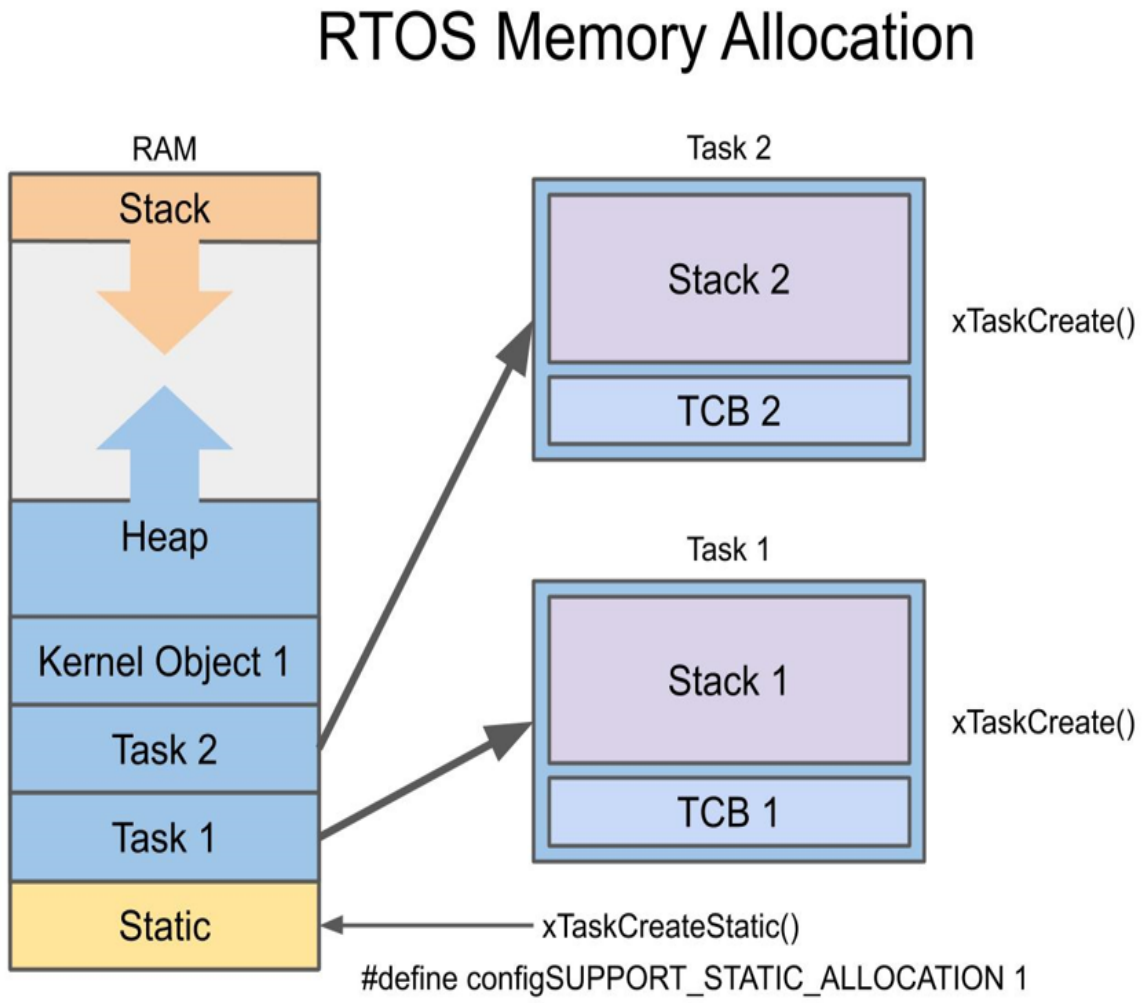
```
int globalVar;  // Uninitialized global variable
static int staticVar;  // Uninitialized static variable
```

```
int initializedGlobalVar = 42;  // Initialized global variable
static int initializedStaticVar = 100;  // Initialized static vari
```



11.2. La zone de tas: Heap:

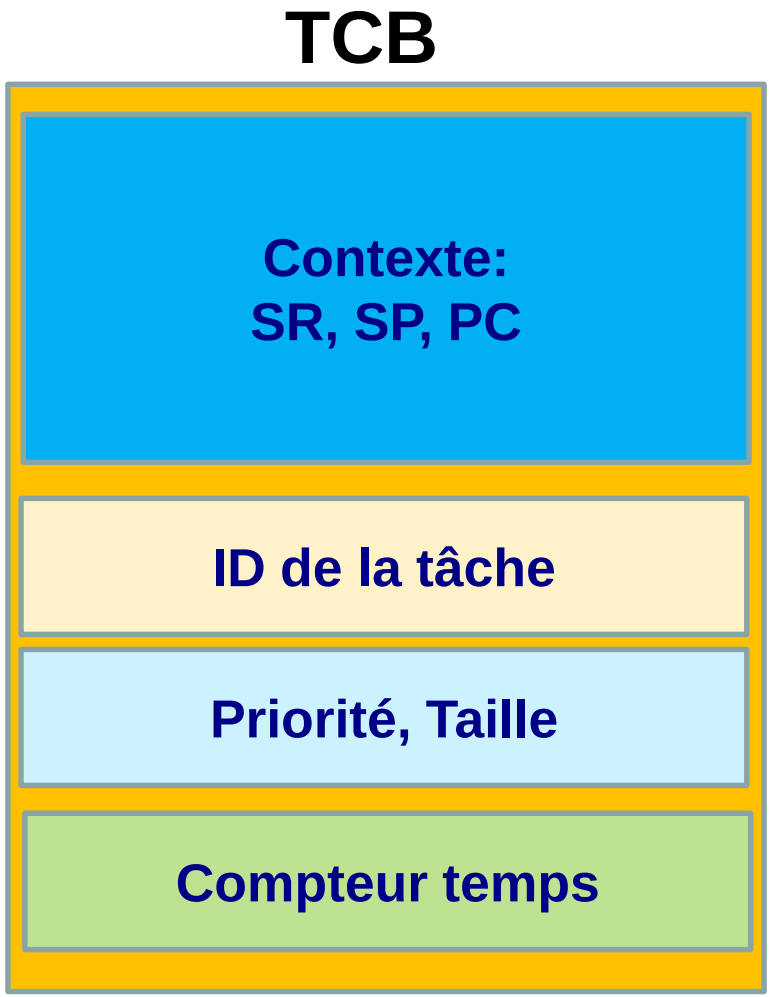
- **Stack: Pile principale:** pour stocker les données et variables temporaires
- **Heap**: « zone de tas »: est utilisée pour l'allocation dynamique de mémoire lors de l'exécution du programme. C'est là que les tâches ou les processus peuvent demander de la mémoire supplémentaire au fur et à mesure de leur exécution.
- La **taille** du Heap est configurable.



11.3. TCB (Task Control Block):

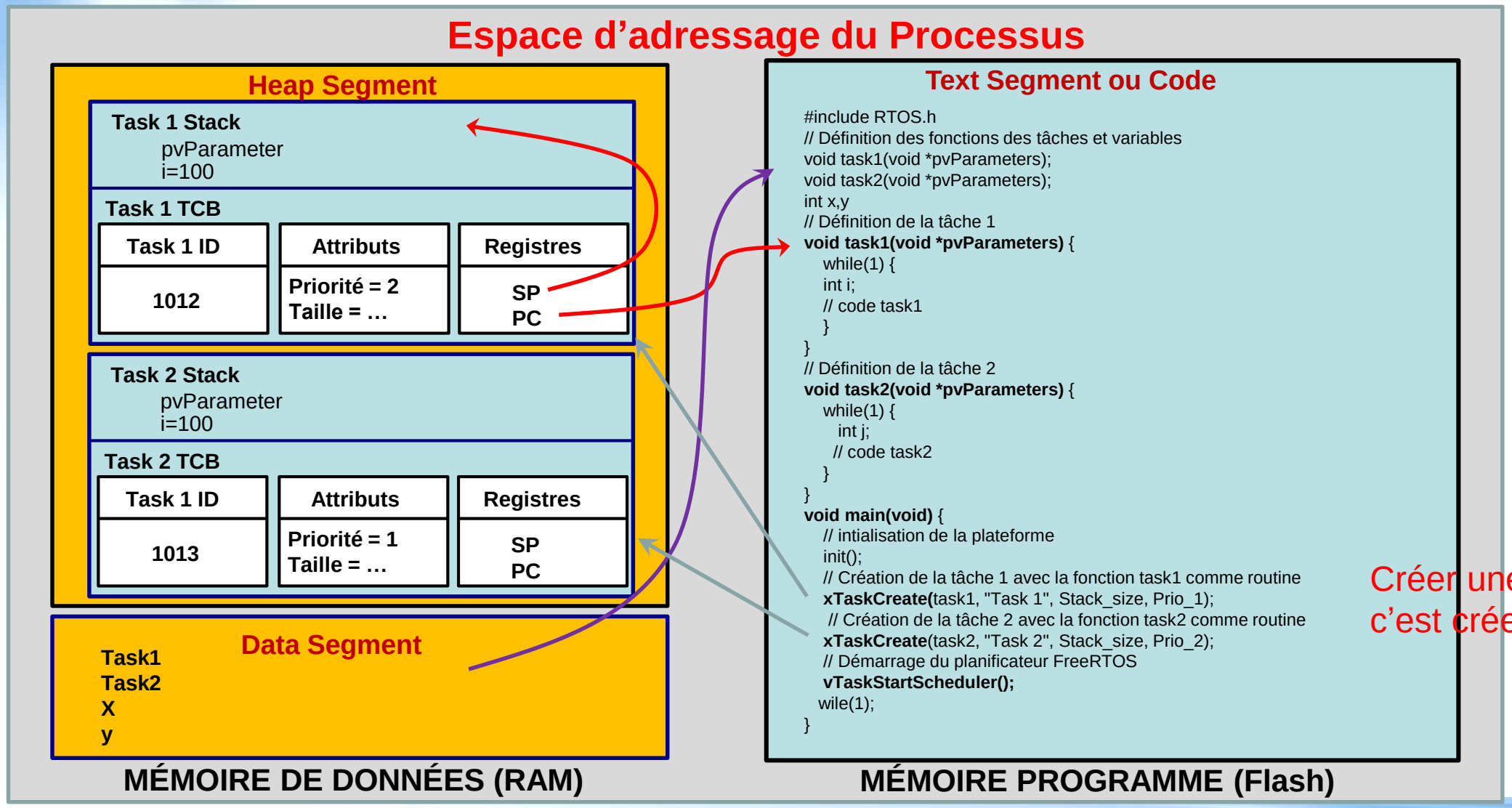
Le TCB contient des informations critiques sur la tâche qui sont nécessaires à sa gestion et à son exécution par le système d'exploitation. Les éléments d'une TCB:

- **Identifiant de la tâche**.
- **Etat de la tâche :** prête, en cours d'exécution, en attente de ressources, suspendue, terminée, etc.
- **Priorité** assignée à la tâche.
- **Contexte de la tâche :** registres CPU associées à la tâche:
  - **SR**: **Registre d'état** (Status Register),
  - **SP**: **Pointeur de pile (**Stack Pointer) : Un pointeur vers la zone de pile (stack) de la tâche
  - **PC**. **Compteur de Programme** (Program Counter)
- **Compteurs de temps :** Dans les systèmes temps réel, des compteurs de temps peuvent être inclus dans le TCB pour suivre le temps d'exécution de la tâche, les délais, les échéances, etc.

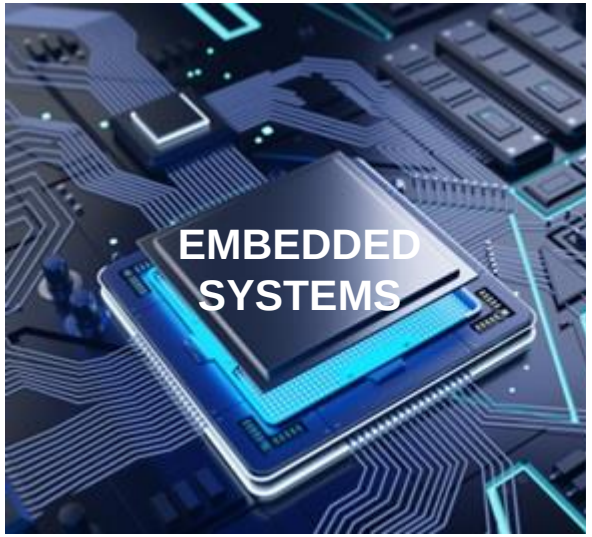


12. Processus

- Un processus est une instance (ou occurrence) d'un programme informatique en cours d'exécution sur un système informatique.
- Chaque processus a généralement ses propres ressources allouées, son espace mémoire, son contexte d'exécution, etc.
- Un processus peut être composé de plusieurs threads (ou tâches dans le contexte des RTOS) qui sont des unités d'exécution plus petites et plus granulaires.
- Les threads partagent souvent les mêmes ressources et l'espace mémoire du processus parent, mais ils peuvent exécuter des parties distinctes du code en parallèle.



Créer une tâche c'est créer son TCB



FIN !
Questions ?