Implémentation d'une Commande numérique en temps réel

Une séance par semaine quinzaine : Jeudi de 11H00 à 12H30

Présence obligatoire

Examen en fin de semestre : X/20

Bonus durant les cours : B

Moyenne = X+B

Implémentation d'une Commande numérique en temps réel

Chapitre I : Description des systèmes temps réel — 2 S

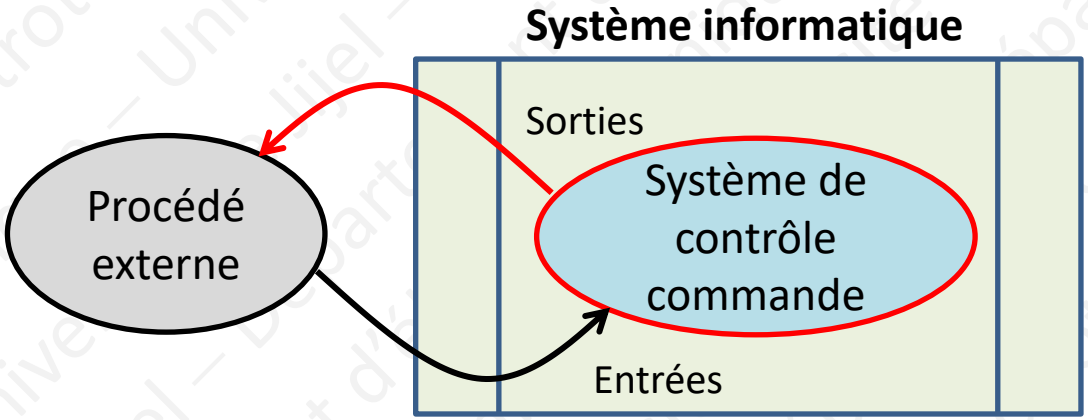
Chapitre II : La commande numérique des systèmes — 2 S

Chapitre III : Etude de l'implémentation des techniques MLI sur un processeur numérique — 4 S

Chapitre IV : Exemples d'implémentation de commandes des machines. Machine à Courant Continu, Machine Asynchrone, Machine Synchrone — 5 S

1. Définitions

Nous pouvons définir un système de contrôle-commande comme un **système informatique** en relation avec l'environnement physique réel externe par l'intermédiaire de capteurs et/ou d'actionneurs. Contrairement aux systèmes d'informatiques scientifiques (gestion de base de données, CAO, bureautique…) qui ont des entrées constituées de données fournies par des fichiers ou éventuellement un opérateur.
Les grandeurs physiques acquises permettent au système de contrôle-commande de piloter un procédé physique quelconque. Donnons ainsi une définition générale d'un système de contrôle-commande.



2. Définitions

Un système de contrôle-commande reçoit des informations sur l'état du procédé externe, traite ces données et, en fonction du résultat, évalue une décision qui agit sur cet environnement extérieur afin d'assurer un **état stable**

Un système informatique de contrôle-commande se différencie des systèmes informatiques classiques par ces deux caractéristiques :

Résultats très dépendant de la vitesse du processeurs : dans un système de traitement informatique classique, le résultat d'un calcul est indépendant de la vitesse d'exécution du processeur. Par contre, dans un système de contrôle-commande, la stabilité du système (résultat) dépend de la dynamique de traitement (vitesse du processeur)

Reproductibilité des résultats: dans un système de traitement informatique classique, le résultat d'un calcul est toujours le même si on lui présente les mêmes donnée. Par contre les données d'un système de contrôle-commande sont issues de capteurs et sont rarement identiques.

3. Caractéristiques

Les contraintes temporelle d'un système de contrôle-commande sont divisées en deux :

contraintes temporelles relatives ou lâches (temps réel mou : *soft real-time*) : les fautes temporelles sont tolérables (ex. : jeux vidéo, applications multimédia, téléphonie mobile…) ;

contraintes temporelles strictes ou dures (temps réel dur : *hard real-time*) : les fautes temporelles ne sont pas tolérables (ex. : avionique, véhicules spatiaux, automobile, transport ferroviaire…).

Les systèmes de contrôle-commande peuvent aussi être classés en trois types:

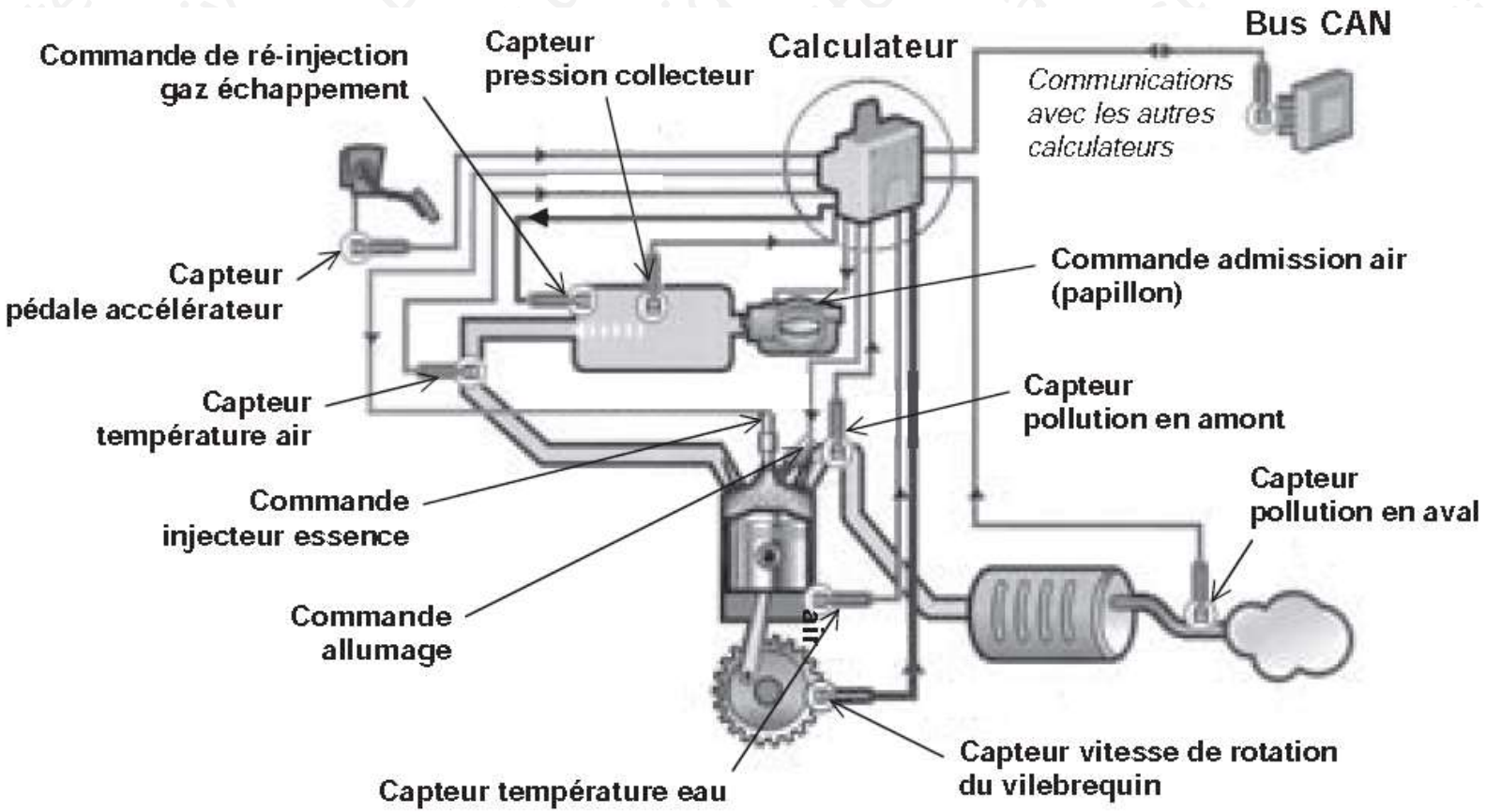
système de contrôle-commande **embarqué** (*embedded real-time system*) : pas d'intervention humaine directe (pas de modification du programme ou des paramètres du programme) ;

système de contrôle-commande **dédié** (*dedicated real-time system*) : les architectures matérielles ou logicielles sont spécifiques à l'application (noyau, processeur…) ;

système de contrôle-commande **réparti ou distribué** (*distributed real-time system*) : l'architecture matérielle est constituée de plusieurs processeurs reliés entre eux par un bus ou un réseau.

2. Caractéristiques

Exemple commande moteur à combustion



2. Caractéristiques

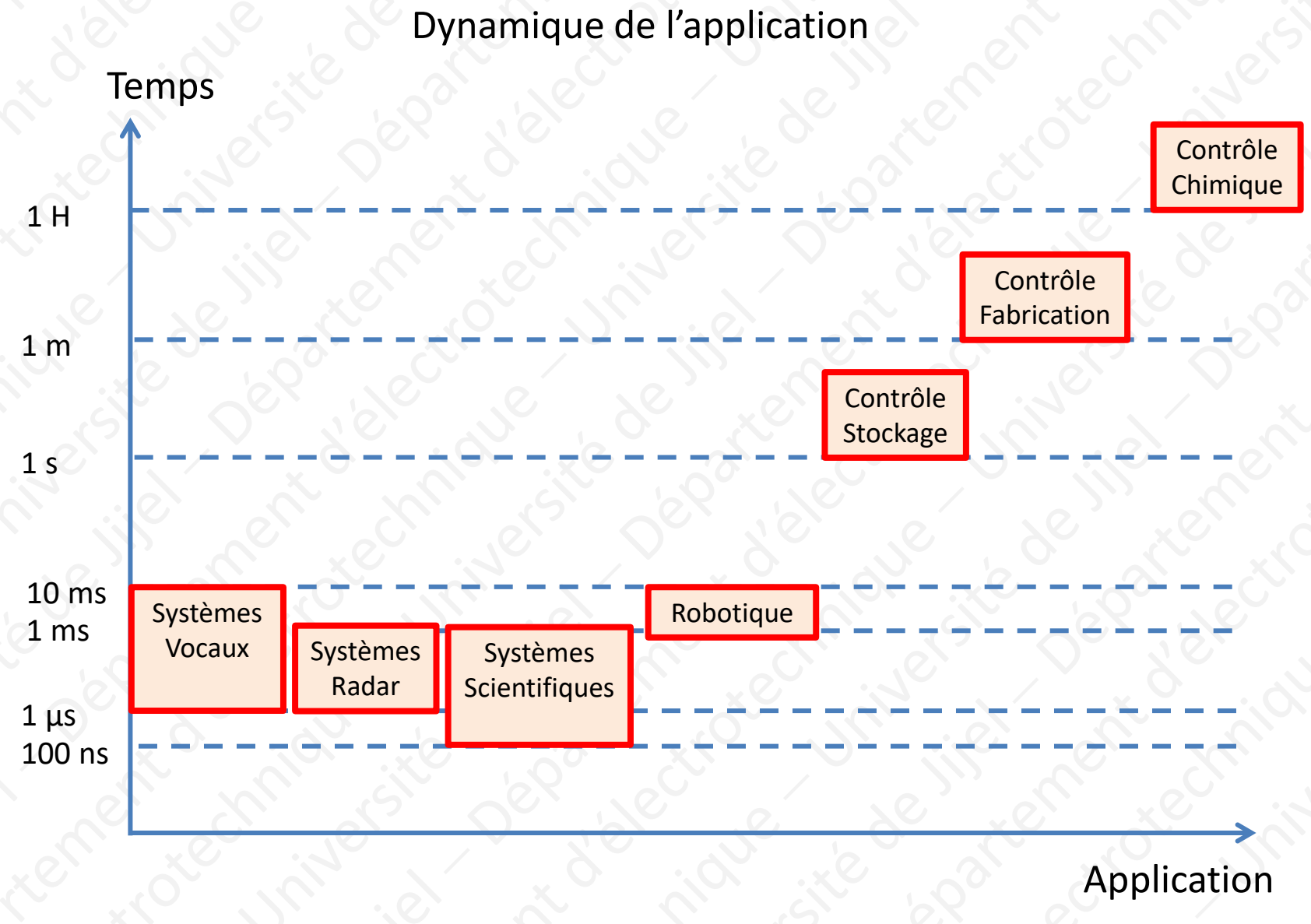
grande diversité des dispositifs d'entrées/sorties : les données à acquérir qui sont fournies par les capteurs et les données à fournir aux actionneurs sont de types très variés (continu, discret, tout ou rien ou analogique). Il est aussi nécessaire de piloter un bus de terrain pour les communications;

prise en compte des comportements concurrents : l'ensemble de ces donnés physiques qui arrivent de l'extérieur et le réseau qui permet de recevoir des messages ne sont pas synchronisés au niveau de leurs évolutions, par conséquent, le système informatique doit être capable d'accepter ces variations simultanées des paramètres ;

respect des contraintes temporelles : la caractéristique précédente impose de la part du système informatique d'avoir une réactivité suffisante pour prendre en compte tous ces comportements concurrents et en réponse à ceux-ci, de faire une commande en respectant un délai compatible avec la dynamique du système ;

sûreté de fonctionnement : les applications qui sont mise en jeu demandent un niveau important de sécurité pour raisons de coût ou de vies humaines. Pour répondre à cette demande, il est nécessaire de mettre en œuvre toutes les réponses de la sûreté de fonctionnement (développements sûrs, tests, prévisibilité, continuité de service, tolérance aux fautes, redondance, etc.).

3. Caractéristiques temporelles



3. Caractéristiques temporelles

Il est nécessaire ici de préciser et de formaliser cette caractéristique temporelle qui peut prendre de nombreuses formulations. On peut citer par exemple :

La durée d'exécution d'une activité : l'activité d'une application, qui peut être l'enchaînement de plusieurs activités élémentaires (acquisition, traitement, commande, affichage…), possède une durée d'exécution qui peut être mesurée de diverses manières. Cette durée n'est pas constante à chaque occurrence de cette activité puisque les programmes et les enchaînements de programmes ne sont pas toujours identiques (branchement conditionnel, itération, synchronisation…)

La périodicité (Cadence de répétition) d'une activité : l'acquisition d'une donnée ou la commande d'un actionneur peuvent nécessiter une régularité liée par exemple à la fréquence d'échantillonnage.

date de réveil : dans certains cas, un traitement doit être déclenché à une date précise relative par rapport au début de l'exécution de l'application ou absolue (plus rarement). Cette date de réveil n'implique pas obligatoirement l'exécution ; il peut y avoir un délai de latence dû à l'indisponibilité du processeur.

3. Caractéristiques temporelles

Date d'activation : cet instant correspond à l'exécution effective de l'activité

échéance (Date au plus tard): le traitement ou la commande d'un actionneur doivent être terminés à un instant fixé par rapport au début de l'exécution de l'application. Dans le cas d'applications à contraintes temporelles strictes, cette échéance doit être respectée de façon impérative, sinon il y a faute temporelle et l'application est déclarée non valide.

Temps de réponse : cette caractéristique peut s'appliquer à une activité de régulation ou à un ensemble d'activités de régulation ; elle est directement liée à la dynamique du système. Ce paramètre correspond à la différence entre la date de réveil et la date de fin de l'activité.

Gigue temporelle : ce paramètre caractérise la répétabilité d'une activité au fur et à mesure de ses occurrences. En effet, entre deux exécutions successives d'une même activité, ses caractéristiques temporelles peuvent changer : date d'activation, durée d'exécution, temps de réponse, etc.

4. Exemples

Robot de production : un robot, réalisant une activité spécifique (peinture, assemblage, tri) sur une chaîne de production, doit effectuer son travail en des temps fixés par la cadence de fabrication. S'il agit trop tôt ou trop tard, l'objet manufacturier traité sera détruit ou endommagé conduisant à des conséquences financières ou humaines graves (oubli d'un ou plusieurs rivets sur un avion).

4. Exemples

Robot d'exploration : ce robot doit se déplacer dans un environnement en principe non connu (zone radioactive après un accident, planète, épave sous la mer…). Il est important qu'il puisse réagir aux obstacles fixes ou mobiles afin de ne pas conduire à sa perte.

4. Exemples

Téléphone mobile : le système de contrôle-commande doit remplir plusieurs fonctions dont certaines ont des contraintes temporelles fortes pour avoir une bonne qualité de service (QoS :*Quality of Service*). Ainsi, la première fonction est de transmettre et de recevoir les signaux de la parole (577 µs de parole émises toutes les 4,6 ms et 577 µs de parole reçues toutes les 4,6 ms à des instants différents).

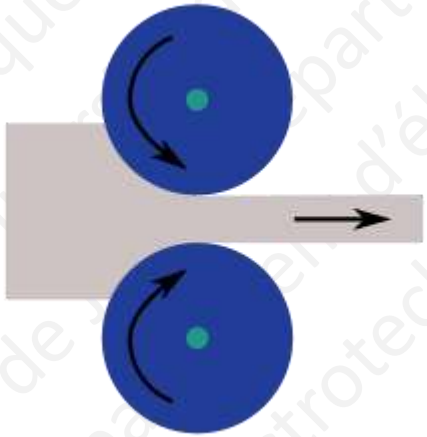
En parallèle, il est nécessaire de localiser en permanence le relais le plus proche et donc de synchroniser les envois par rapport à cette distance (plus tôt si la distance augmente et plus tard si la distance diminue). Des messages de comptes rendus de la communication sont aussi émis avec une périodicité de plusieurs secondes. Les contraintes temporelles imposées au système doivent être imperceptibles à l'utilisateur..

4. Exemples

Système de vidéoconférence : ce système doit permettre l'émission et la réception d'images numérisées à une cadence de 20 à 25 images par seconde pour avoir une bonne qualité de service. Afin de minimiser le débit du réseau, une compression des images est effectuée. D'autre part la parole doit aussi être transmise et doit être synchronisée avec le flux d'images. L'ensemble de ces traitements (numérisations et compression d'images et parole, transmission, réception, synchronisation…) sont réalisés en cascade, mais avec une cohérence précise.

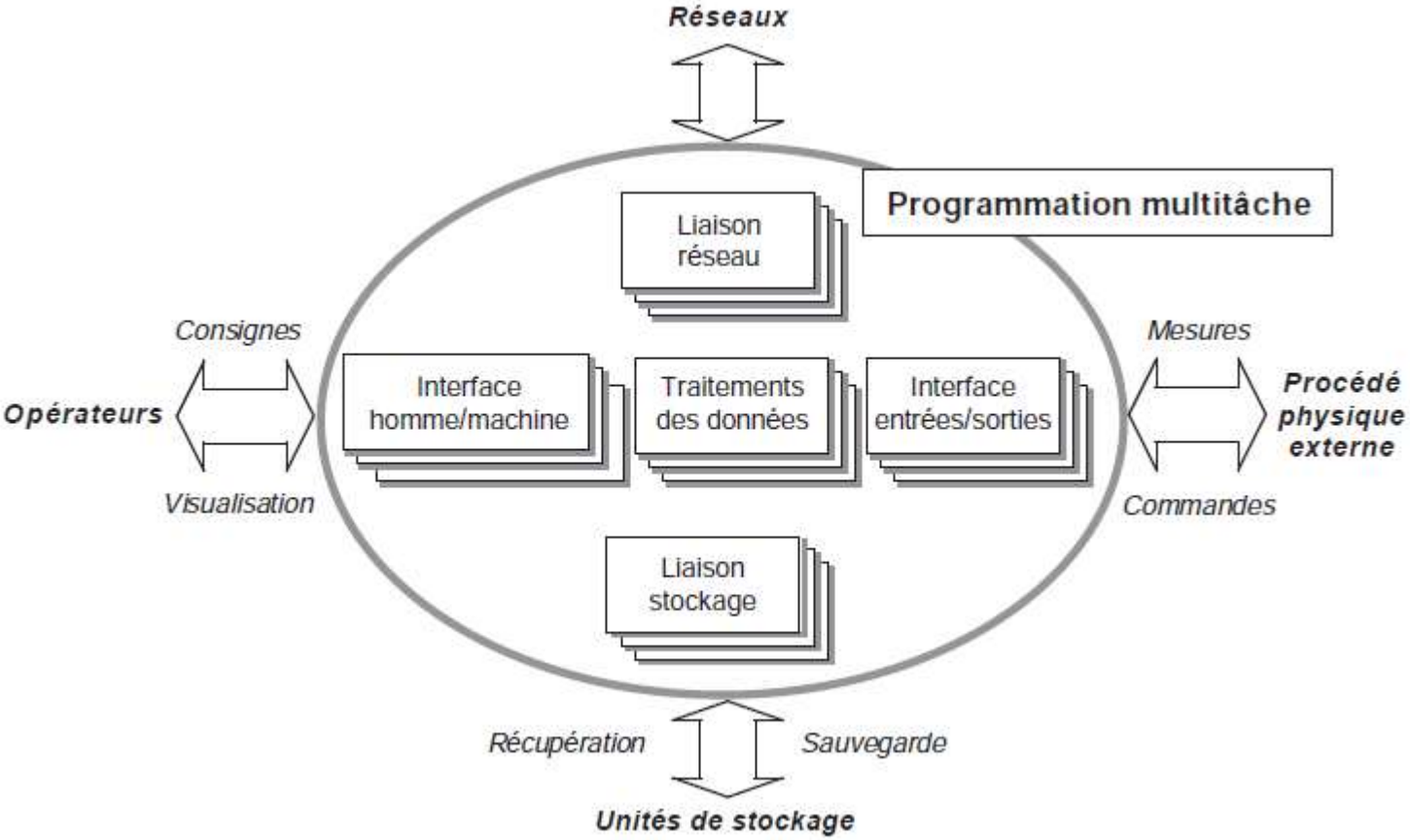
4. Exemples

Pilotage d'un procédé de fabrication (fonderie, laminoir, four verrier…) : par exemple la fabrication d'une bobine d'aluminium (laminage à froid) exige un contrôle en temps réel de la qualité (épaisseur et planéité). Cette vérification en production de la planéité nécessite une analyse fréquentielle (FFT) qui induit un coût important de traitement. Le système doit donc réaliser l'acquisition d'un grand nombre de mesures (246 Ko/s) et traiter ces données (moyenne, FFT…) à la période de 4 ms. Ensuite, il affiche un compte rendu sur l'écran de l'opérateur toutes les 200 ms et enfin imprime ces résultats détaillés toutes les 2 s. Un fonctionnement non correct de ce système de contrôle de la qualité peut avoir des conséquences financières importantes : production non conforme à la spécification demandée.



Architecture logicielle des systèmes temps réel.

Le système physique à commander possède des événement concurrents de caractéristiques temporelles différentes, ce qui nous amène à considérer ce système comme ayant un comportement fortement parallèle. C'est pour cette raison que la conception de la partie logiciel doit suivre ce **comportement parallèle** et doit donc être **multitâche**.



Architecture logicielle des systèmes temps réel.

Tâches d'entrées/sorties : ces tâches permettent d'accéder aux données physiques à travers de cartes d'entrées/sorties, de capteurs et d'actionneurs directement liés au procédé géré. Ces tâches peuvent être activées de **façon régulière** ou par **interruption**.

Tâches de traitement : ces tâches constituent la partie principale de l'application. Elles intègrent des traitements de signaux (analyse spectrale, traitement d'images, etc.) ou des lois de commande (régulation tout ou rien, régulation du premier ordre, régulation PID, etc.).

Tâches de gestion de l'interface utilisateur : ces tâches permettent de présenter l'état du procédé ou de sa gestion à l'utilisateur. En réponse, l'opérateur peut modifier les consignes données ou changer les commandes. Ces tâches peuvent être très complexes et coûteuses en temps de calcul si l'interface gérée est de taille importante (tableau de bord) ou de type graphique (représentation 3D).

Tâches de communications : ces tâches sont destinées à gérer les messages envoyés ou reçus à travers un ou plusieurs **réseaux** ou **bus de terrain**. Si ce type de tâches existe, l'application est dite distribuée ou répartie.

Tâches de sauvegarde : ces tâches permettent de stocker l'état du système à des instants fixés. Cette sauvegarde peut être utilisée *a posteriori* pour analyser le fonctionnement de l'application ou lors d'une reprise d'exécution à une étape précédente.

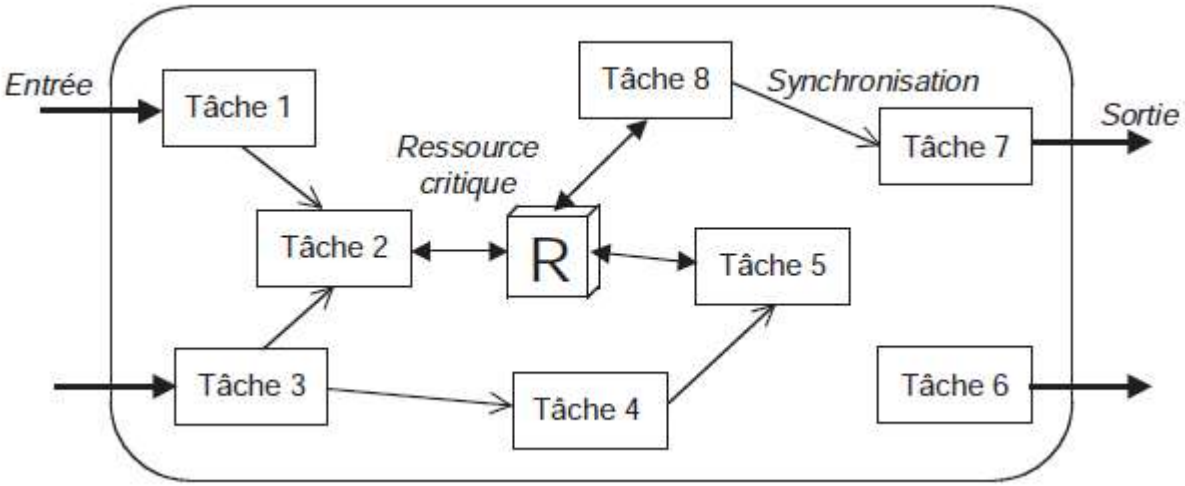
Architecture logicielle des systèmes temps réel.

Ces tâches peuvent être liées par des relations de différents types :

Synchronisation : les tâches sont reliées par une relation de précédence d'exécution

Communication : ici les tâches partagent un certain nombre de données entre elles, il peut y'avoir aussi une relation de synchronisation.

Partage de ressource : les tâches utilisent les éléments mis en commun dans le système (mémoire, entrées sortie, réseau,…). Certaines ressources ne doivent pas être accessible en même temps par plusieurs tâches ; ce sont des ressources critiques (la mémoire par exemple)



Architecture logicielle des systèmes temps réel.

Cette architecture logicielle est donc un ensemble de tâches concurrentes, synchronisées, communicantes et partageant des ressources critiques. Le rôle du système informatique est donc de gérer l'enchaînement et la concurrence des tâches en optimisant l'occupation du processeur, cette fonction est appelée **ordonnancement**

Il existe deux types d'ordonnancement : **synchrone** et **asynchrone**

Exemple : soit une application de contrôle qui utilise deux tâches:

Lecture_Consinge : tâche qui permet de lire les consignes d'un opérateur entrée à l'aide d'un clavier. Cette tâche prend un certain temps à cause de l'intervention humaine.

Alarme : tâche qui se déclenche suite à événement extérieur critique. Cette tâche doit être prise en compte au plus vite pour éviter l'endommagement du procédé

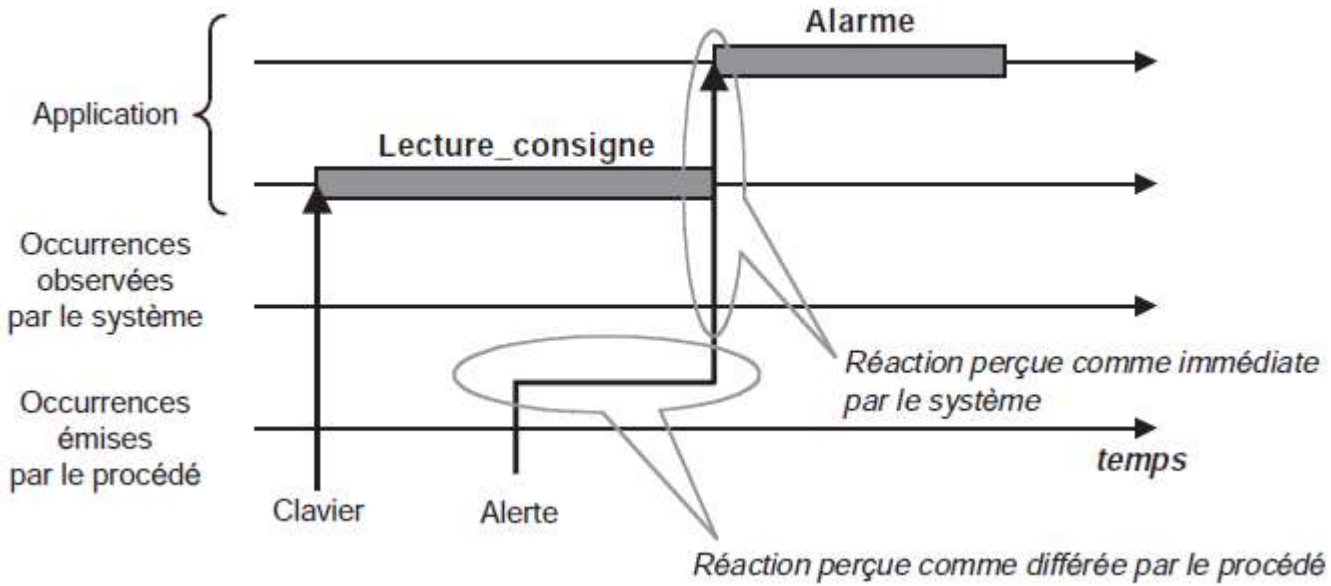
On suppose que la tâche Lecture_Consigne s'exécute en premier

Architecture logicielle des systèmes temps réel.

Dans le modèle d'exécution synchrone, la tache Lecture_Consigne masque la tâche Alarme jusqu'à la fin de son exécution.

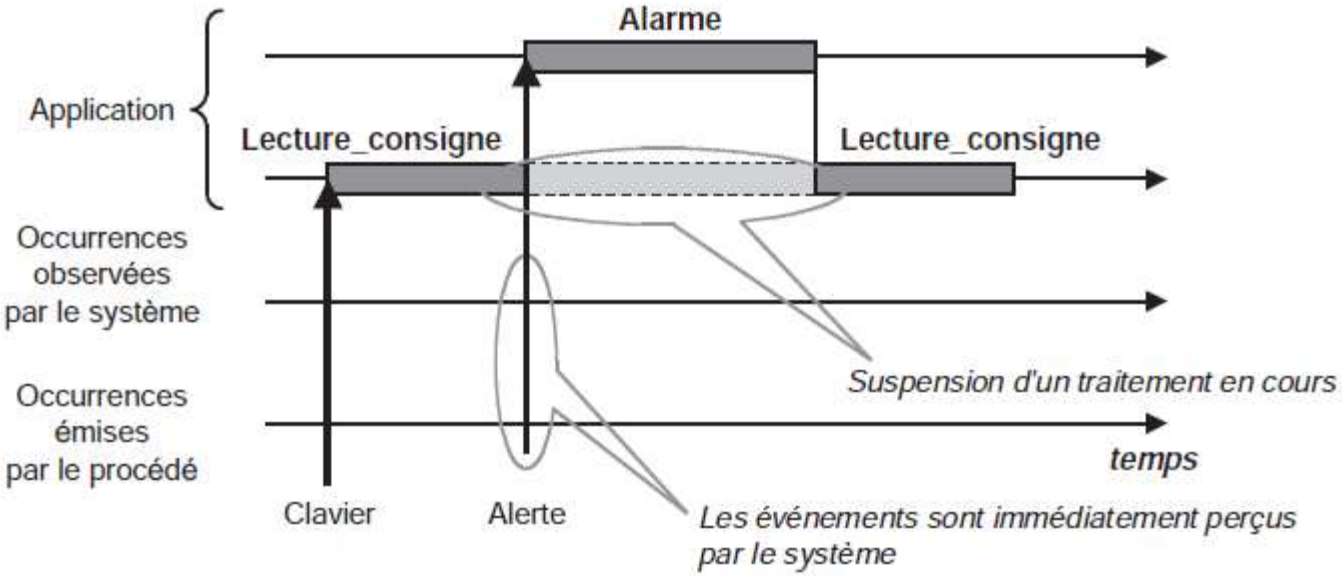
Pour le système la tâche alarme est prise en compte immédiatement (après la tâche Lecture_Consigne)

Pour le procédé la tache Alarme est retardé de l'évènement déclencheur.



Architecture logicielle des systèmes temps réel.

Dans le modèle d'exécution asynchrone, la tache Lecture_Consigne est suspendue pour permettre l'exécution de la tâche Alarme,
Dans ce cas les événements extérieur seront pris en compte suivant un ordonnancement qui dépend de l'urgence ou de l'importance de chacun



Architecture logicielle des systèmes temps réel.

Il y a donc à retenir les notions suivantes :

Préemptibilité

- une tâche préemptible est une tâche que l'on peut suspendre à n'importe quel moment et affecter le processeur à une autre. Dans ce cas la prise en compte des évènements extérieurs peut être très courte mais la programmation doit prendre en considération les mécanismes de partage de ressources critiques.
- Une tâche non préemptible, est une tâche qui ne peut être interrompue que par elle-même et à des endroit précis (fin de tâche, attente….). Dans ce cas la programmation devient simple (pas de partage de ressources) mais les évènements extérieurs risque d'être retardés

Analyse de l'ordonnancement

- L'analyse de l'ordonnancement hors ligne correspond à la construction d'une séquence d'exécution complète sur la base des paramètres temporels des tâches. L'ordonnanceur nécessaire est minimal puisque la séquence d'exécution est prédéfinie, il se réduit à un séquenceur. En revanche, l'application ainsi figée est peu flexible.
- L' analyse de l'ordonnancement en ligne correspond à un choix dynamique de la prochaine tâche à exécuter en fonction des paramètres de la tâhe. L'ordonnancement a un coût temporel non négligeable ; en revanche, l'application peut réagir à des événements ou des situations non prévus.

Architecture logicielle des systèmes temps réel.

Synchronisme

- Une exécution est dite synchrone si les tâches sont non préemptibles et s'exécutent les unes après les autres dans un ordre qui peut être défini par une analyse hors ligne de l'ordonnancement.

- Une exécution est dite asynchrone si les tâches sont préemptibles et s'exécutent selon l'ordonnancement. Une analyse de la séquence doit se faire obligatoirement en ligne.

Architecture matérielle des systèmes temps réel.