

Temps-Réel Libre et Conduite d'un Robot Mobile Opto-guidé

Maryline Chetto* , Christophe Plot**

* Institut de Recherche en Communication et cybernétique de Nantes UMR CNRS 6597 BP 92101 - 1 Rue de la Noé, F-44321 Nantes cedex 3 maryline.chetto@univ-nantes.fr

** Centre de Ressources Technologiques et de Transferts Industriels IUT de Nantes, Avenue du Professeur Jean Rouxel BP , F-44475 Carquefou cedex Christophe.plot@univ-nantes.fr

RÉSUMÉ : Le travail présenté ici a été réalisé dans le cadre d'un projet¹ national de R&D dans le domaine des technologies logicielles dédiées aux applications temps-réel, initialisé en 2002. Ce projet avait pour finalité la conception d'un système d'exploitation temps réel baptisé « système Cléopâtre » à code source ouvert et gratuit. Ce système d'exploitation temps réel a la particularité de se présenter sous forme d'un assemblage de composants sélectionnables par l'utilisateur selon ses besoins. Nous présentons le démonstrateur de ce projet, un AGV (Automated Guided Vehicle) conduit en temps-réel par l'O.S. Cléopâtre. L'objectif était d'évaluer ses performances d'un point de vue temporel, le gain engendré par l'approche composant en termes d'empreinte mémoire et d'accessibilité mais aussi vérifier l'inter-opérabilité des composants logiciels développés. L'application mise en œuvre consiste à conduire cet AGV opto-guidé muni de son informatique embarquée, en exécutant en temps-réel des ordres de mission émis par une liaison sans fil à partir d'une station de supervision.

MOTS-CLÉS : temps-réel, chariot optoguidé, système d'exploitation, logiciel libre.

1. Linux et le temps-réel

Depuis plusieurs années déjà, Linux conquiert un domaine où il n'était pas particulièrement attendu, celui des applications temps réel. Et si l'on retrouve ce système d'exploitation généraliste dans le domaine très spécifique de l'informatique temps réel, c'est qu'on lui reconnaît des qualités autres que celles reconnues dans l'environnement standard type PC grand public, à savoir : sa gratuité, l'ouverture de son code source, les nombreuses distributions proposées, l'aide rapide par la communauté Internet des développeurs...

Linux possède en effet de nombreux autres atouts pour son utilisation dans le secteur des systèmes temps réel embarqués : son portage sur processeurs autres que 80x86 tels que PowerPC, ARM ou MIPS, une taille du noyau modeste compatible avec la taille des mémoires utilisées dans les systèmes embarqués (< 500 Ko), un support pour le chargement dynamique de modules qui permet d'optimiser la taille du noyau, une migration rapide et en douceur pour un spécialiste Linux à Linux Temps Réel, réduisant ainsi les temps de formation et par voie de conséquences les coûts... A mi-chemin entre les développements maison effectués sur mesure en interne, et ceux qui se construisent autour de systèmes d'exploitation et de plates-formes propriétaires, Linux ouvre donc une nouvelle voie.

¹ Projet financé par le Ministère de la Recherche, notification n° 01 K 0742

C'est donc dans cette nouvelle voie que nous avons inscrit la conception et le développement du système d'exploitation Cléopâtre, extension temps réel de Linux.

Linux est à l'origine un système généraliste élaboré pour le temps partagé, et non pour le temps réel. Deux approches permettent d'octroyer à Linux des spécificités temps réel. Une première idée consiste à modifier directement le noyau en remplaçant son ordonnanceur natif par un ordonnanceur préemptif à priorité qui améliore son comportement. Ce type de solution convient bien aux applications dites temps réel à contraintes légères pour lesquelles la performance s'évalue en termes de temps de réponse moyen. Cependant, lorsque les contraintes temps réel sont strictes, un noyau entièrement dédié aux applications temps réel devient indispensable. La solution consiste alors à ajouter un noyau auxiliaire, permettant de relayer les processus non temps réel à un niveau de priorité le plus bas. Les promoteurs de cette technologie sont ceux qui considèrent que le noyau Linux ne sera jamais véritablement temps réel et lui ajoute des fonctionnalités dont certainement le plus fondamental est un ordonnanceur déterministe. Cette technologie est utilisée par RTLinux et son semblable européen, RTAI, depuis la fin des années 90 (Mantegazza P., 1999).

2. Le projet Cléopâtre

Les applications temps réel s'avèrent de plus en plus complexes et ne sont pas, soit à contraintes strictes, soit à contraintes légères : elles mettent en œuvre des tâches dont les contraintes sont hétérogènes et leurs caractéristiques temporelles de plus en plus variées. Leur respect ne peut être garanti ni par un système temps réel classique (à priorité fixe calculée sur la période par exemple) comme Linux/RTAI ni par un système d'exploitation dit à temps partagé conventionnel comme Linux.

Les travaux de recherche menés ces dernières années ont d'ailleurs prouvé, par validation théorique des résultats, la nécessité de se tourner vers des ordonnanceurs adaptatifs et flexibles, à priorité dynamique (Earliest Deadline) (Liu C. et al. 1973) pour gérer des tâches apériodiques, faire face à des situations de surcharge temporaire de traitement par la garantie d'une qualité de service minimum, mettre en œuvre des tests d'admission en ligne, etc.

L'objectif du projet RNTL (Réseau National des Technologies Logicielles) Cléopâtre initialisé en 2001 et officiellement terminé en 2005 était donc de faire la preuve de l'intégrabilité de fonctionnalités novatrices dans un système d'exploitation ouvert tel que Linux et ce, sous forme de composants sélectionnables à la carte en fonction du profil de l'application (Chetto M. et al., 2004).

Voulant s'affranchir du développement complet d'un noyau temps réel juxtaposé à Linux et de son interface matérielle, il s'est naturellement fait jour que la meilleure solution était de concevoir Cléopâtre comme un patch à une extension temps réel existante de Linux, qui se présenterait sous la forme d'une librairie de composants logiciels munie de son interface.

S'inscrivant dans ce point de vue, l'ensemble des membres du consortium Cléopâtre a ainsi pris l'option de considérer RTAI comme support de développement avec les principales motivations suivantes : projet communautaire, contributeurs majoritairement européens, excellente documentation, nombreuses architectures supportées, licence LGPL et non GPL, gestion du temps réel dur dans l'espace utilisateur... On peut alors considérer Cléopâtre comme un patch de RTAI, lui-même patch de Linux.

3. Un O.S. temps-réel à la carte

Nous avons fourni sous forme de composants à code source ouvert, des fonctionnalités de niveau noyau innovantes en matière de tolérance aux fautes (mécanisme à échéance et calcul imprécis), de synchronisation (protocoles d'accès aux ressources critiques avec héritage de priorité et priorité plafond) et d'ordonnancement (ordonnancement à priorité dynamique, serveur de tâches aperiodiques, tests d'admission en ligne de tâches critiques) (voir Figure 1). Nous avons démontré l'applicabilité et l'interopérabilité des composants par des tests en simulation puis sur des applications réelles mises en œuvre par les différents partenaires de notre consortium et en particulier sur une application de robotique mobile décrite ci-dessous.

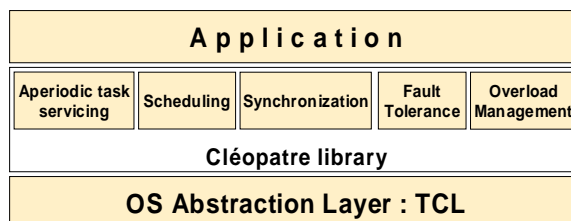


Figure 1. La librairie de services temps-réel

Cette gamme de composants se répartit sur plusieurs étagères de la bibliothèque et est suffisamment étendue pour toucher un panel le plus large possible de secteurs d'applications, allant des applications les plus critiques, en général à caractère fortement cyclique (industrie militaire, avionique) aux applications les moins critiques (multimédia, vidéo à la demande) en passant par des applications où cohabitent des contraintes à degré de criticité échelonnés (robotique mobile avec télé-opération).

Nous avons doté le système Cléopâtre de propriétés d'extensibilité qui lui permettent de s'adapter aux futurs besoins des industriels et aux nouvelles idées des chercheurs par le biais de développement de nouveaux composants interopérables (Marchand A. et al., 2006). Les composants d'une même étagère ont la propriété de s'inter-changer, ce qui laisse les utilisateurs libres de choisir le mécanisme le plus adapté à leurs besoins parmi plusieurs ou d'en tester de nouveaux sans modifier leurs applications.

Les développements de composants Cléopâtre pour plusieurs configurations système et matérielles sont factorisés par une couche de contrôle des tâches (TCL) qui en fait abstraction (voir Figure 2).

Ainsi les composants sont génériques et le système s'adapte à un nouvel environnement pour un investissement réduit. D'ailleurs une couche de contrôle des tâches a déjà été développée en quelques mois pour la version 3.0 de RTAI qui ne repose plus sur le patch RTHAL qui donne le contrôle du système Linux à un noyau temps réel, mais sur le patch ADEOS qui est un pipeline de systèmes d'exploitation.

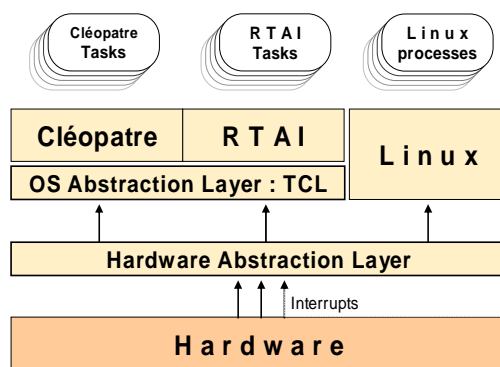


Figure 2. : Architecture logicielle

4. Le démonstrateur : un AGV

La phase de prototypage en charge de valider le travail scientifique du projet CLEOPATRE avait comme objectif :

- d'expérimenter sur une application réelle, l'O.S. temps-réel à architecture configurable,
- de montrer son adéquation aux exigences des développeurs.
- d'illustrer pour un intégrateur potentiel, le coût global d'utilisation de ces composants lors du développement, de l'exploitation et de la maintenance d'une application.

Le CRTTI de Nantes, partenaire du consortium Cléopatre a conçu un robot mobile autonome permettant de mettre en place une multitude de scénarii qui testent et certifient les composants tant du point de vue de la sûreté de fonctionnement (fiabilité, sécurité) que du respect des contraintes temporelles.

Le robot mobile est un AGV qui réalise des missions en milieu industriel transmises par liaison radio sous forme d'ordres simples (voir Figure 3). Pour cela, il progresse le long d'une piste matérialisée sur le sol par un ruban adhésif de couleur et est muni d'un tapis roulant servant à charger et décharger des bacs standards d'usine (Dixon J et al., 1997). La piste est balisée par des réflecteurs catadioptriques qui marquent les emplacements où le robot doit exécuter un ordre.

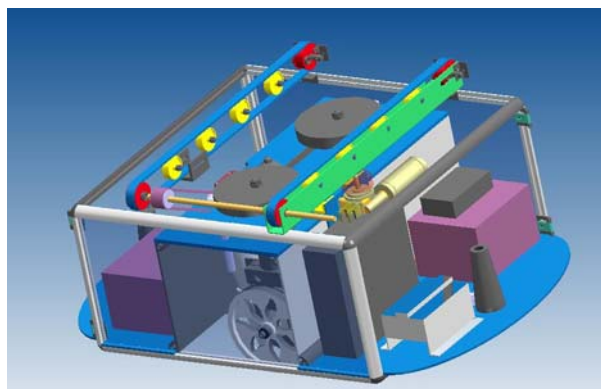


Figure 3. La plate-forme mobile

4.1 Instrumentation du robot

La propulsion est réalisée à l'aide de deux moteurs de traction contrôlés par des variateurs de vitesse en mode *RI* (CM1 et CM2). Deux moteurs de direction (CM3 et CM4) dirigent les roues. Ces moteurs peuvent être bloqués pour offrir la possibilité d'utiliser le robot en mode holonome ou non. Son tapis roulant est actionné par un dernier moteur (MT) (voir Figure 4). Le robot mobile est pourvu de photo-résistances permettant de réaliser le suivi de la piste (SdP1 et SdP2), de capteurs de présence de bacs (PB1 et PB2), de capteurs infrarouges TOR détectant les catadioptres présents sur la piste (PA0 à PA2) et de capteurs de proximité détectant les obstacles (DIR1 à DIR4).

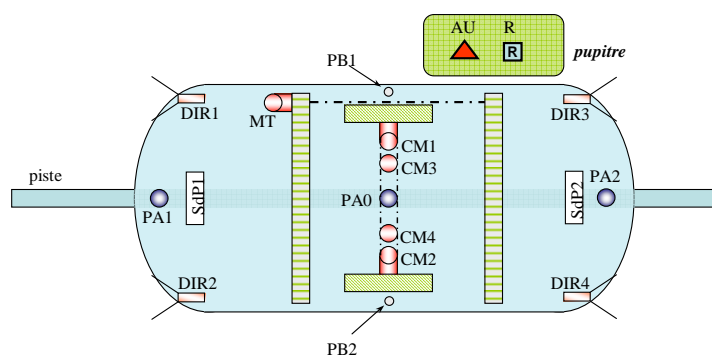


Figure 4. Capteurs du robot mobile

Le robot est doté d'un pupitre de commande avec un bouton d'arrêt d'urgence (AU) et un bouton de réarmement (R) après un arrêt d'urgence. La liaison radio est assurée par un modem correspondant à la norme 802.11.

Le robot mobile est contrôlé par PC industriel embarqué PC104 relié au matériel par une carte d'entrées/sorties PC104-AIO12-8 avec 8 entrées analogiques, 4 sorties analogiques, 24 entrées/sorties numériques et 3 compteurs/timers. Une mémoire compact flash de 32 Mo joue le rôle de disque dur. La mémoire compact flash contient le système Cléopâtre et l'application qui contrôle le robot selon les ordres transmis par liaison radio.

4.2. Implémentation

Le logiciel d'application se décompose en trois tâches aperiodiques dédiées à la réception de nouveaux ordres, à l'émission de messages de progression du robot, et à l'exécution des ordres. Cette dernière tâche coordonne d'autres tâches dédiées aux différents ordres que le robot mobile est susceptible d'exécuter : *suivre la piste, tourner à gauche, tourner à droite, faire demi-tour, charger un bac, suivre une direction pour un aiguillage...* Ces tâches agissent sur deux tâches de commande qui appliquent chacune une vitesse à une des roues du robot mobile (voir Figure 5).

Par exemple, la tâche de *suivi de piste* est chargée de faire progresser le robot le long de la piste. Elle utilise les données fournies par un système matériel de suivi de piste connecté aux photo-résistances (SdP1 et SdP2) qui déterminent si le robot se trouve à gauche, à droite ou au centre de la piste. Ces données sont appelées *erreur*. A partir de cette *erreur*, la tâche de *suivi de piste* calcule une *correction d'erreur* qu'elle transmet aux tâches de contrôle des roues pour rectifier la trajectoire du robot et éviter qu'il quitte la piste.

Deux stratégies de calcul de la *correction d'erreur* ont été envisagées afin de valider les mécanismes de tolérance aux fautes du système Cléopatre. Il s'agit de la *correction proportionnelle* (rapide mais peu précise) et de la *correction proportionnelle intégrale* (plus lente mais précise). Toutefois, des mesures ont montré un écart temporel insuffisant entre les deux stratégies pour valider les composants de tolérance aux fautes du système Cléopatre.

Le robot possède également des capteurs de proximité permettant de détecter les obstacles. Ces capteurs génèrent une interruption dont la routine réveille la tâche de priorité maximale qui est chargée d'immobiliser le robot dans les plus brefs délais. Un *système de retour vidéo* a également été développé pour qu'un opérateur suive le déplacement du robot à l'aide de caméras embarquées à l'avant et à l'arrière du robot.

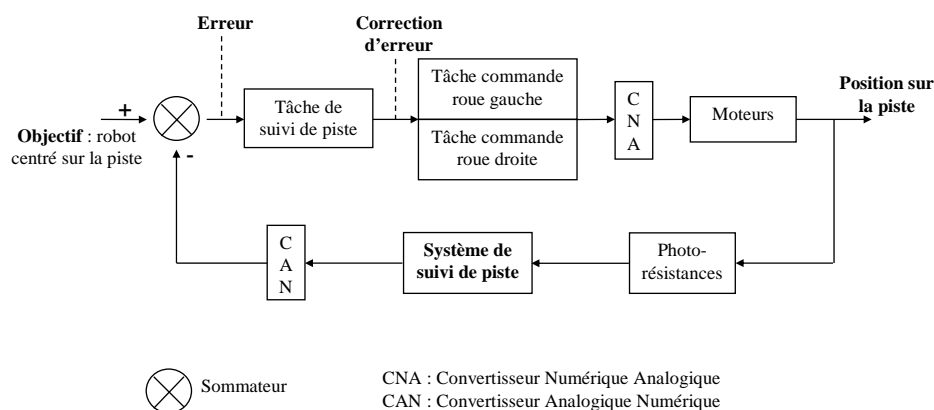


Figure 5. Schéma-bloc du suivi de piste

L'O.S. Cléopatre et l'applicatif ont été installés sur 2 Mo de la mémoire compact flash. En effet, le chargement à la carte des composants logiciels système se caractérise par une très faible empreinte mémoire, critère de performance primordial pour les systèmes embarqués mais également critère économique.

Cléopatre a été développé en mettant l'accent sur l'amélioration de l'accessibilité de Linux temps réel. Pour cela, il met à disposition de nombreuses fonctionnalités qui simplifient le travail du programmeur et réduisent le temps de prise en main du système. Remarquons que ce travail d'intégration sur l'AGV a été réalisé en moins de six mois par un stagiaire élève-ingénieur sans connaissance préalable de ce système d'exploitation. Des outils novateurs tel que l'arrêt d'urgence logiciel, les zones de non préemption, les signaux d'abandon des tâches, etc. sont autant d'outils d'aide au développement dont l'utilité a pu être démontrée par le biais de cette application.

5. Conclusion

Les travaux menés sur cette plate-forme mobile ont permis de réaliser deux objectifs :

- évaluer le système d'exploitation temps-réel Cléopatre et la plupart de ses composants logiciels. Une application robotique offre en effet la particularité de faire cohabiter des tâches périodiques tournant à des cadences élevées (asservissement) et des tâches événementielles déclenchées lorsque surviennent des événements extérieurs (informations

capteurs, commande opérateur). Cette cohabitation de ces deux types de tâche a été une bonne validation pour les ordonnanceurs et les serveurs de tâches aperiodiques et vérifier l'inter-opérabilité de ces services offerts sur des étagères différentes de la librairie.

- Démontrer les avantages de cette approche « OS temps-réel à la carte » en terme de gain de temps de développement et facilité d'utilisation dans le domaine du génie industriel.

6. Références

Chetto M. and Garcia-Fernandez T., (2004) Enhancing Linux/RTAI with Open Source Software Components, 6th Real Time Linux Workshop, Singapour.

Chetto M. (2006), <http://cleopatre.rts-software.org>.

Dixon J, Henlich O., (1997) Mobile Robot Navigation. Londres : Imperial College. Course for Information Systems Engineering. Available from internet:<http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/jmd/>

Liu C., Layland J.W., (1973) Scheduling algorithms for multiprogramming in a hard real-time environment .Journal of ACM, vol.20 n°1 pp.46-61.

Mantegazza P., (1999) DIAPM RTAI for Linux: Why's, what's and how's, Real Time Linux Workshop, University de Technology of Vienna.

Marchand A. and Chetto M., (2006) “Dynamic Real-Time Scheduling of Firm Real-time Tasks with Hard andSoft Aperiodic Tasks, The Journal of Real-Time Systems , Kluwer Academic Publishers.