CLEOPATRE: FREE OPEN-SOURCE COMPONENTS FOR REAL-TIME CONTROL OF ROBOTIC APPLICATIONS

M. Silly-Chetto [1], P. Bonnin [2], P. Blazevic [3], F. Russotto [4] , C.Plot [1], V. Dupourqué [5], P. Pomiers [5] and R. Aubin [3]

[1] University of Nantes, La Chantrerie BP 50609 Nantes Cedex 03 France
[2] L2TI, University of "Paris Nord", 93430 Villetaneuse
[3] Laboratoire de Robotique de Versailles, 10-12 Avenue de l'Europe, 78140 Vélizy France
[4] CEA/FAR 18, Route du Panorama, 92265 Fontenay-aux-Roses, France
[5]ROBOSOFT, Technopole Izarbel, F-64210 Bidart

Abstract: This paper is concerned with a software framework for real-time computing which supports a component based system design targeted to robotic applications. We are providing a set of free open-source software components that enhance LINUX/RTAI by supplying it with a library that contains both system-level modules including dynamic schedulers and Fault-tolerance mechanisms, and user-level modules dedicated to real-time vision and automatic control.

Keywords: Real-time, Software architecture, Open-source components, Robotics.

1. INTRODUCTION

In this paper, we describe a national project[*] that aims at the improvement of embedded computing systems for robotic applications with hard real-time constraints. This project, initialized at the end of 2001 carries out research and technology development in collaboration with four academic partners from France. The issues of test and demonstration are adressed by two industrial partners.

Systems integrators today need to build larger and more sophisticated systems with less time, money and resources. Nowadays, the development remains an often badly mastered aspect, notably by misunderstanding of the "real-time" concepts. So, the essential motivation is built on the necessity, for integrators, of having solutions which improve, first the productivity and the evolutivity of the developed application software and second, the predictability of the resultant system. As seen by the increasing number of research and industrial projects dealing with the issue, it seems that Linux is in the way to become an appealing platform for developing real-time systems, in particular in the robotic domains.

The objective of the project is first to participate in the evolution of an opened community standard, Linux and second to provide a set of open source free software components for developing embedded applications especially targeted to robotics.

The key objective is as well to demonstrate the applicability and the interoperability of these software components under a free open-source real-time operating system, Linux/Rtai (Garcia, 2003). The design of our component-based framework is divided in two parts. The first part is concerned with system-level components that contain real-time facilities such as schedulers, aperiodic task servers and fault-tolerance services. The second part contains application-level components dedicated to real-time vision and automatic control. The latter ones will use the former ones. This library is consequently offered at the Rtai-Linux level as independent kernel modules and as user-level functions. These components will be made available mainly under the GNU Lesser General Public License.

The tests on a mobile robotic platform (an Automated Guided Vehicle) are performed to show the benefits in terms of both improved integration process and adequacy with strict requirements on safety and reliability of next generation applications.

2. REAL-TIME LINUX

In the wide scenario of the embedded systems market, free open-source operating systems are going to represent an effective alternative to the two

[*] Work supported by the French Research Office, grant number 01 K 0742.

common approaches: “in-house” and commercial. This effectively will allow the developers to concentrate their efforts where they are most productive i.e on developing applications.

It has been now ten years that GNU tools are used for embedded software. The unmistakable advantage of this model lies in the earning in flexibility and the possible option to modify the code in the optics to answer specific needs. Regrettably, neither the various versions of Real-Time Linux (RTAI, RT-Linux, Hard Hat Linux, Red Linux, KURT), nor the proprietary executives do answer suitably the much diversified requirements posed by the new applications. These products do not offer either library of application-level services. They propose relatively basic kernels. RTAI offers the same services of the Linux kernel, adding the features of a real time operating system. RTAI mainly works as an interrupt dispatcher that intercepts the peripheral interrupts and routes them to either Linux or to real-time tasks. RTAI gets all the control over the hardware platform and considers Linux as background task running when no real-time task occurs. The fifos allows the communication among RTAI tasks and Linux processes.
RTAI was chosen because it appeared well documented. Moreover, it permits to patch its components that are disseminated under LGPL license. However, the standard RTAI-Linux has drawbacks and limitations. The RTAI pre-emptive scheduler works with static priorities. Both aperiodic and periodic tasks can be used but the priority of a periodic task bears no relation to its period. Deadlines are not used, and provisions for resources are not present.

3. SYSTEM-LEVEL COMPONENTS

3.1 Dynamic scheduling

One of the key aspects in real time systems concerns the task scheduling policies. A strong work has been done during the last years in static and dynamic scheduling theory, see (Liu, 2000). Whereas static scheduling has been incorporated in the commercial operating systems due to its simplicity, dynamic scheduling has been relegated to a research framework.
A well-known fixed priority scheduling algorithm is the *deadline-monotonic* algorithm (DM) which executes first the task with the shortest relative deadline (Audsley et al, 1993). DM is optimal among fixed priority schedulers. The *Earliest Deadline* algorithm (EDF) executes tasks in the order of their absolute deadline (Liu and Layland, 1973). EDF is optimal among all scheduling algorithms. By the criterion of schedulable utilization, EDF outperforms DM. However, the advantage of DM is the predictability of which tasks will miss their deadline during an overload.
In a hard real-time system, it is necessary to perform an off-line test (a *schedulability test*) for the purpose of validating that the application can meet all its hard deadlines according to a chosen algorithm. When the tasks are independent, sufficient schedulability tests exist both for EDF and DM. These two schedulers have been integrated as selectable components in the library.

3.2 Aperiodic task servicing

An *aperiodic* task is a soft non periodic tasks while a *sporadic* task is a non periodic one with a hard deadline. The objective of the scheduler is:

- To decide whether to accept or reject the newly occurred sporadic task and to schedule the task with all the others.
- To complete each aperiodic task as soon as possible without causing periodic tasks and accepted sporadic tasks to miss their deadlines.

Three alternative approaches have been considered for implementation. According to the *background* server (BGS), aperiodic and sporadic tasks are scheduled and executed only at times when there is no periodic task ready for execution. According to the *Total Bandwidth Server* (TBS), the server has a fixed size budget and fixed period but the server deadline is dynamically adjusted so as to replenish its budget early (Spuri and Buttazzo, 1996).
The *Slack Stealing* approach, named *EDL server* (EDLS) consists in dynamically computing slack in order to keep track of the amount of available processor time for executing the aperiodic tasks as soon as possible and guaranteeing the deadlines of sporadic tasks whenever possible (Chetto and Chetto, 1989). BGS, TBS and EDLS have been integrated as components in CLEOPATRE.

3.3 Fault-tolerance

Even when carefully designed a real-time system can be subject to disturbances which are the effects of subtle errors in software coding, malfunctions in input channels,… The unpredictable occurrence of these disturbances appears as timing failures and consequently results in danger to human life or damage to equipment. It is necessary to provide any real-time operating system with techniques which are able to make the results available on a timely basis even if this one leads to a functioning in " degraded mode ". This is a quality pre-required by the embedded real-time systems.

Time redundancy is an effective method in order to maintain the properties of correctness and timeliness of a real-time system even in the presence of faults. Among the error recovery methods, we have considered the *Deadline Mechanism* where each fault-tolerant task is characterized by two different copies (*primary* and *backup*) (Liestman and Campbell, 1986). The aim of the scheduler is to guarantee the execution of the backup process whenever the primary one fails. The processor time reserved for the execution of the backup process is reclaimed if the primary process executes successfully. Such a fault-tolerant scheduler has been integrated in Linux/RTAI.

3.4 Resource access control

A resource allocation policy is a set of rules that govern when and under what conditions each request for resource is granted and how tasks requiring resources are scheduled. Classical techniques based on semaphores may lead to permanent blocking deadlocks. To prevent them, the library provides additional techniques. The simplest way is to schedule all critical tasks non-preemptively which corresponds to execute the task holding a resource at the highest priority. We call this protocol the *Super-Priority Protocol* (SPP). According to *Priority Inheritance Protocol* (PIP), any task that holds a resource and provokes blocking of another task inherits the priority of the blocked task until it releases the resource. The *Priority Ceiling Protocol* (PCP) is an extension of PIP to prevent deadlocks and reduce the blocking time. It makes use of a parameter associated to every resource, called ceiling priority. See (Sha et al, 1990) for more details.

3.5 Implementation considerations

One important objective of the project is to build components as generic as possible. So, we isolated the links concerning task management (creation and destruction), context switches and interruptions from the operating system. The specificities of the operating systems are gathered in a special structure called TCLCreateType, which can be transmitted through intermediary modules in order to reach the operating system.
The Task Control Layer (TCL) keeps up to date the system clock, elects the tasks of the new system according to their dynamic priority parameter and gives a basic interface to the scheduling components of the system. TCL is the keystone of the system architecture. All source codes developed for CLEOPATRE can be downloaded from the web site of the project at http://www.cleopatre-project.org.

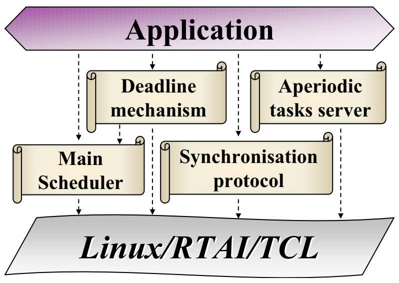

Fig. 1: System-level components in CLEOPATRE

4. APPLICATION-LEVEL COMPONENTS

The target application chosen to validate CLEOPATRE developments involves control of a redundant robot arm and a holonomic mobile robot using a real-time vision system for localization, navigation and path planning. Application-level components have been designed to carry out the services required by the chosen application.

4.1 Real-Time Vision

The fault-tolerance mechanisms are very interesting to design Dynamic Real-Time Vision Systems which adapt their behavior to the immediate available computing power.
Color region feature information is required for the navigation of the demonstrator in order to detect obstacles and to localize itself using colored landmarks (see part 5). But, if edge feature information is also available, the accuracy of the 3D information will be better.
Robust color region information can be extracted using region growing procedures. But, such kinds of procedures are very time consuming (Bonnin and al, 2003). A region growing procedure is used in “normal mode”.
Color Classification followed by an Extraction of Connected Components procedures deliver very rapidly the same kind of information, but less robust. Indeed, the obtained color regions widely depend of the lighting conditions. So these procedures are used for the “degraded mode”. If the information of the region growing procedure is not available at time, they are replaced by those of the “degraded mode”.
At the opposite, if time is available before the deadline, it is possible to add new visual information as those coming from edge detection.

4.2 Automatic control

Developments for robot-arm control are based on the redundant manipulator Mitsubishi PA10 with 7 DOF[1].
First of all, an I/O management system was needed. An arcnet driver was designed from scratch to allow communications between the computer and the robotic arm.
At this stage, the existing driver is not yet generic because the application is protocol dependant. Nevertheless, the design of this driver gives a basic structure on which we can start a new IO driver design. With this driver we introduced articular control of the PA10, i.e. we designed a simple control loop with proportional gain and speed limits for each of the 7 articulations.
Our next purpose was to provide calculation solutions in order to achieve control in Cartesian space. Thus, we implemented linear algebra functions for:

- matrix manipulations (on rows, columns or elements),
- vector/vector operations,
- matrix/vector operations.

With these functions we implemented the Greville algorithm to compute the pseudo-inverse of a non-square matrix. The pseudo-inversion of a non-square matrix (the Jacobian's transformation matrix in PA10's control) is necessary in an application with redundancy if the robot has to be controlled in Cartesian space.

[1] DOF - Degree of freedom

At this level, we introduced a more complicated control loop which includes calculations and communications.
We designed in parallel a real-time simulation composed with both real-time module for control and graphical user interface using OpenGL/GLUT.

Developments for mobile-robot control are based on the holonomic mobile base developed by the CRTTI (see demonstrator section).

These developments are based on an existing CEA industrial robotic application which has been under development for several years and is used in most of CEA robotic products. The industrial application is implemented for the VxWorks RTOS and one of the works carried out for the CLEOPATRE project was porting the application components to RTAI + CLEOPATRE.

Application components have been redesigned to be as much modular and generic as possible in order to enhance reusability and portability. For instance, a generic RT component has been specifically redesigned to provide a standardized API encapsulating some common features mainly used in the context of a robotic application. By locating RT specific features in a dedicated area, this component dramatically enhances application portability towards several RT OS or kernels (eg: RTAI+CLEOPATRE, RTAI, RT-Linux, VxWorks …) and eases application developers work. This component will encapsulate CLEOPATRE specific features in order to demonstrate scheduling policy efficiency enhancement and to provide target application useful fault-tolerance mechanisms to be used for safety recovery in case of application fault.

Several components are now ported and unitary tested. These components provide: common linear algebra for robotic control (matrix, vector, quaternion, force/torque and jacobian computation), common control toolboxes (signal processing, control-laws …) and some other common generic toolboxes. Next development steps will consist in designing the holonomic mobile robot model, a trajectory generator for such mobile robot, a control mode supervisor, I/O management and a communication component allowing to remotely control the mobile robot. For test purposes and prior to integration on the mobile platform, a mobile-robot simulator will also be developed in cooperation with the CRTTI.

5. DEMONSTRATOR

An industrial mobile robot has been built in order to test the library of modules with a real application. The bi-directional mobile robotic platform is fitted out with 2 conveyor belts in order to carry standard vats. It is an holonomous robot with two driving wheels (4 DOF). The mechanical support includes the following components: frame, batteries, motor reducers, wheels, the system of vats loading and the embedded PC. Moreover, the follow-up of track, the pace monitoring and the obstacle and collision detection are based on various embedded sensors.

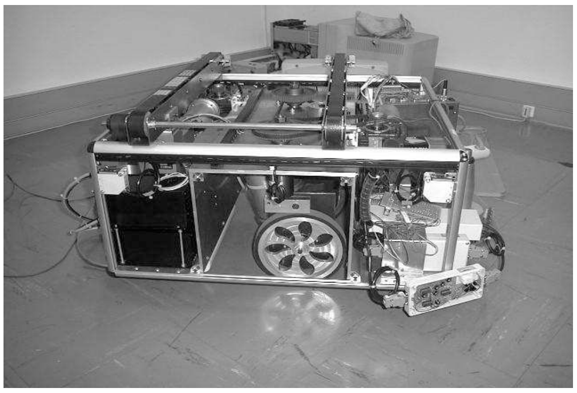
Fig 2 : Holonomous robot

The robot supports 2 guidance modes:

- The *guided* mode when the robot is required to follow the track adjusting its trajectory according to the measured variation.
- The *blind* mode when the robot deviates from the track.

First, depending on the direction of walk, one of the two embedded track sensors is used to provide the error corresponding to the variation detected.

These two sensors provide an analogical signal ranging between -10V and +10V, corresponding to the variation measured compared to the track, in terms of distance. An illustration is given below :

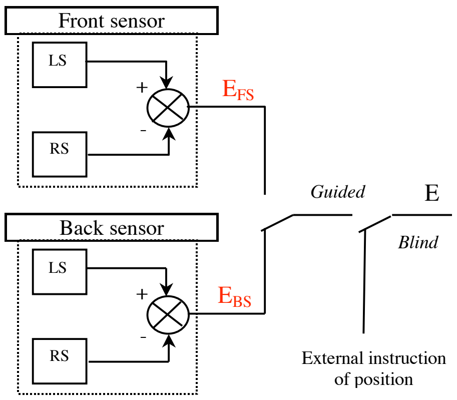

Fig. 3: Follow-up of track error obtaining

Then the error is brought back in a definite interval, adjusted by a regulator K and added or withdrawn from the instruction speed so as to worked out the orders of respectively the right and left amplifiers. Thus, the engines'control is carried out in a completely symmetrical way.

In this paper, we shall consider *the follow-up of track control* problem for the system. More precisely, we shall deal with the control system such that the resulting closed-loop system is stable.

Initially, we implement a simple proportional regulator :

$$U(z) = K_P E(z)$$

where E(z) and U(z) are respectively the control input and the controlled output.

The results are not optimal ones. Indeed, oscillations testify a certain instability of the system :

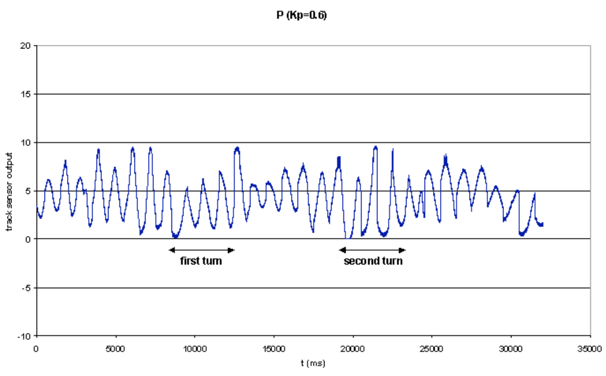

Fig. 4: System behavior with a P regulator

In order to obtain the best performance from the system, the use of a more elaborate device, namely a Proportional Integral regulator (PI) is preferable.

While the proportional term is used to adjust the speed of the system, the integral control is used to provide the required accuracy for the control system. For the taking into account of the integral term, it is necessary to perform and add a digital Integral controller in the preceding form (4.1) for the system :

$$u(k+1) = u(k) + \int_{kT}^{(k+1)T} e(\tau)d\tau$$

$$= u(k) + \frac{T}{2}\left[e(k+1) + e(k)\right]$$

$$zU(z) = U(z) + \frac{T}{2}\left[zE(z) + E(z)\right]$$

$$U(z) = \frac{T}{2}\frac{z+1}{z-1}E(z)$$

Then, the resulting equation for the digital PI controller is :

$$U(z) = \left[Kp + Ki\frac{T}{2}\frac{z+1}{z-1}\right]E(z)$$

Our approach was to use a technique which was developed in the 1950' s but which has stood the test of time and is still used today. This is known as the Ziegler Nichols tuning method. The calculated values were not optimal values and additional fine tuning was required to obtain the best performances from the system.

This demonstrator enables us to demonstrate the impact of real-time technology in industry where safety and reliability are so important. A first phase in our experimentation was to control the robot using the native Linux/RTAI Operating System with fixed priorities. At present, the second test phase consists in measuring performance gain (in term of deadline guarantee) obtained by integration of the new libraries here described.

6. CONCLUSIONS AND FUTURE WORK

Today, the main challenge of the project is to demonstrate the efficiency of this new Linux-based operating system. A performance evaluation study is conducted all the overheads induced both by interrupt latency and context switch and by the different mechanisms which are implemented in the module

This work is a first step towards a larger open software platform which could be shared by the community of Advanced Robotic Applications developers, in which different robots models and simulation systems should be added, also as open source software and modules.

A preliminary public web site in French has been set-up and can be visited at : http://www.project-cleopatre.org.

REFERENCES

N. Audsley, A. Burns, K. Tindell, M. Richardson and A. Wellings (1993), Applying a new scheduling theory to static priority preemptive scheduling. In *Software Engineering Journal***,** Vol. 5, No. 5, pp. 284-292.

H. Chetto and M. Chetto (1989), Some results of the Earliest Deadline scheduling algorithm. In *IEEE Trans. On Soft. Eng.,* Vol. 15, No. 10, pp. 1261-1269.

T.Garcia, A.Marchand and M.Silly-Chetto (2003), CLEOPATRE: A R&D Project for Providing New Real-Time Functionalities to Linux/RTAI. In *Proc. of Fifth Real-Time Linux Workshop*, Valencia, pp. 119-124.

A.L. Liestman and R.H. Campbell (1986), A fault-tolerant scheduling problem. In *IEEE Trans. On Soft. Eng.,* Vol. 12, No. 11, pp. 1089-1095.

C.L. Liu and Layland (1973), Scheduling algorithms for multiprogramming in a hard real-time environment. In *J.Assoc. Comput. Mach.*, Vol.20, pp. 46-61

L. Sha, R. Rajkumar and J.P. Lehoczky (1990) Priority inheritance protocols: An approach to real-time synchronization. In *IEEE Trans. on Computers***,** Vol. 39.

M. Spuri and G. Buttazzo (1996), Scheduling aperiodic task scheduling for hard real-time systems. In *Real-Time Systems Journal,* Vol. 10, pp. 179-210.

P.Bonnin, O.Stasse, V.Hugel, P.Blazevic, G.Dauphin, Towards a method to compare and to evaluate fast pixel gathering mechanisms for real time robotic vision systems. *In Proc. of ISSPA 2003*, Paris, July 2003, pp 157-160