Xenomai Lab

A Platform for Digital Real-Time Control

Jorge Amado-Azevedo[1], Alexandre Mota[1], and Paulo Pedreiras[2]

[1] Instituto de Engenharia Electrónica e Telemática de Aveiro, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, 3810, Portugal
[2] Instituto de Telecomunicações, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, 3810 Aveiro, Portugal

Abstract. Most real-time Linux solutions have seen fairly low adoption among control engineers. The difficulty of use and specific computer science knowledge required have overshadowed the fact that many of these solutions are free and open-source software.
Xenomai Lab is a free Linux application which attempts to bridge this gap by allowing control systems to be modeled in a visual form using block diagrams. Each block represents a control algorithm written in C which may be edited by the user.
The resulting system runs in real-time with the enhanced performance offered by the Xenomai framework. No prior knowledge of Xenomai is needed as all calls to the the underlying system services are hidden from the user. This makes the application adequate for use by control engineers without a background in real-time systems, as well as by entry level students of control engineering, robotics and related areas.

Keywords: Linux, Xenomai, Real-Time, Control

1 Introduction

Xenomai is a real-time Linux framework under active development since 2001 [1]. It is part of a family of Linux variants such as RTAI [2] and RTLinux [3] which attempt to transform Linux into a real-time operating system (RTOS) – an OS with deterministic real-time behavior and support for real-time semantics [4].

RTOSs are commonly used by control engineers to implement digital control systems. Temporal determinism and operating reliability are essential for processes such as sampling and actuation [5]. Usually, however, the RTOSs used are not based on Linux, but rather on proprietary technology.

The lack of appeal of Linux might be attributed to numerous reasons. For one, most Linux real-time solutions require a significant understanding of RTOS and computer science concepts such as kernel compilation, real-time threads,

This work was partially supported by the Portuguese Government through FCT - Fundao para a Ciência e a Tecnologia, in the scope of project Serv-CPS-PTDC/EEA-AUT/122362/2010

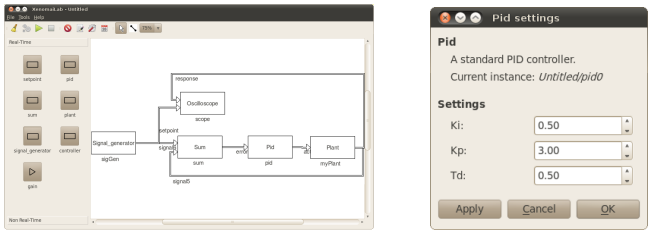

Fig. 1. Screenshots showing XL's main window (left) and a block GUI (right)

inter-process communication, etc. Proprietary alternatives offer simpler solutions, which greatly improves the productivity of the user.

Another possible reason is the lack of support for visual modeling. Real-time Linux usually implicates programming a monolithic program in C. This is contrary to how control systems design is usually done, with a diagram representing the system. A C program de-emphasizes dataflow and does not expose internal signals, which makes them cumbersome to visualize in real-time.

Xenomai Lab (XL) is an attempt to bring this type of visual and RTOS-concept free modeling to Xenomai [6]. The application presents the user with a list of blocks and a canvas on which to place and inter-connect them, as seen in Figure 1 (left). Each block in the diagram represents an algorithm which the user can edit at any time. Additionally, each block has an associated GUI that allows run-time tuning of algorithm parameters. An example is show in Figure 1.

Together, these elements describe a Multiple-Input Multiple-Output control system which can then be executed and fine-tuned in real-time.

The approach taken is novel as, while the model of the system is visual, the actual programming is done directly in C. The most common alternatives, such as MATLAB/Simulink or SciCos/RTAI, rely on high level programming and generating C code to a real-time target.

Xenomai Lab's focus of development was ease of use and extensibility. Some of the features include real-time task management being transparent to the user, built-in tools for data visualization, built-in support for matrices and extensive documentation covering both usage and installation.

In the remainder of this paper we will cover XL's implementation, present an example and, finally, draw some conclusions.

2 Implementation Overview

The basic structure of a block is illustrated in Figure 2. Each block is a regular Linux process containing a real-time Xenomai task responsible for applying the control algorithm on the inputs and outputting the result.

A connection between 2 blocks assures that the output of one is fed to the input of the other. As each block is a different process, the data exchange requires the use of Inter-Process Communication (IPC) mechanism.

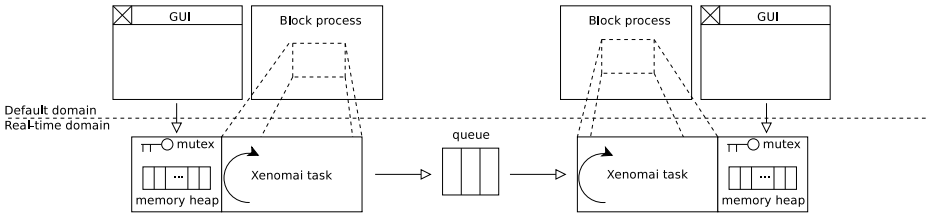

Fig. 2. Block implementation

The Xenomai API offers several tools for real-time IPC. One such tool is a *queue* [7]. A queue is a piece of memory which *producers* use to write values to, and *consumers* use to read values from. These roles are not mutually exclusive and any number of processes or tasks may interact with a given queue.

As can be seen in Figure 2, blocks use queues to communicate. This is adequate for two reasons. Firstly, the queue's producer/consumer semantic is a natural fit for the problem – one block produces a value which another consumes. Secondly, read calls on a queue may be blocking. This allows us to implement block diagrams as a daisy chain of blocks, each one blocked on a read call, waiting for new information to manipulate and pass forward.

The fine-tuning of control algorithm parameters is done by a separate GUI application. This is another IPC issue, as the GUI must somehow communicate with the block that a parameter has changed value.

Since parameter updates occur sparsely and their values must remain constant between updates, the most adequate IPC mechanism to communicate between GUIs and blocks is shared memory.

The Xenomai API implements this IPC mechanism as a *memory heap* [7]. A memory heap is a memory segment which may be shared across any number of real-time tasks. Figure 2 shows how the GUI and the block use a heap to create what is essentially a set of global variables between processes.

Access to the heap is not done directly. Instead, a real-time mutex protects it from concurrent access. The block acquires the lock before entering the control algorithm and releases it afterwards. This guarantees that parameters don't change mid-algorithm execution, as this would compromise the results.

Blocks are programmed in plain C, while GUIs are Qt projects programmed in C++ [8]. A supporting C library implements all the necessary IPC functionality and hides the details from the user. A base class for the GUIs is also provided that hides the complexity and implements handlers for the different types of parameters (string, integer, double, etc).

XL itself is nothing more than an abstract way of interacting with blocks. Instead of manually maintaining lists of queues, XL does all the bookkeeping automatically in a visual and interactive manner. The application presents the user with a blank canvas, and allows him to drop blocks and connect them using the mouse. It also provides convenient ways to edit, compile, execute and monitor block execution.

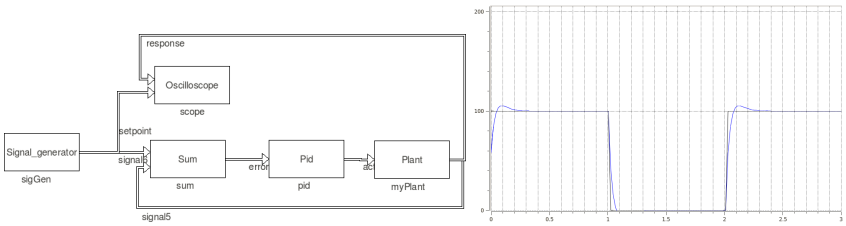

Fig. 3. Plant in a feedback loop with PID controller (left) and response (right)

3 Example

Figure 3 shows a control system using a PID controller for a plant with transfer function $H(q) = \frac{0.0952}{q-0.9048}$ with $h = 1ms$. For the PID gains shown in Figure 1, the plant's step response was greatly improved from the original low-pass behavior.

4 Conclusion

The purpose of this article has been the presentation of Xenomai Lab as a valuable contribution to the world of real-time Linux. The application leverages existing technology to enable a workflow familiar to control engineers. The application has been released under the GPL license, an early alpha release of which is available at `www.xenomailab.org`.

The approach taken by XL lends itself naturally to a pedagogical setting. In the Technical University of Wroclaw, Poland, a teaching package featuring XL has been prepared. The project has several pandaboards (see `www.pandaboarg.org`) running Ubuntu 11.04 and challenges students to use them to control simple experimental setups. XL is used to help students design the system and visualize real-time data acquisition. Classes are to start in September 2012.

References

1. Yaghmour, K.: Building embedded Linux systems. O'Reilly Series. O'Reilly (2003)
2. Cloutier, P.e.a.: Diapm-rtai position paper. RTSS 2000 - Real Time Operating Systems Workshop (2000)
3. Yodaiken, V., et al.: The rtlinux manifesto. In: Proc. of the 5th Linux Expo. (1999)
4. et al., J.S.: Tutorial on hard real-time systems. IEEE CSP (1988)
5. Ogata, K.: Discrete-time control systems. Prentice Hall (1995)
6. Amado-Azevedo, J.: Xenomai lab - a platform for digital real-time control. Master's thesis, DETI – Universidade de Aveiro (December 2011)
7. Gerum, P.: A tour of the native api. `http://www.xenomai.org` (2006)
8. Blanchette, J., Summerfield, M.: C++ GUI programming with Qt 4. Prentice Hall in association with Trolltech Press (2008)