Hardware-in-the-loop Environment for Control Systems evaluation under Linux/RTAI

LUIS GARCIA, MANUEL J. LOPEZ, JOSE LORENZO
Dpto. Ingeniería de Sistemas y Automática
Universidad de Cádiz
SPAIN

Abstract: - In this paper we propose a general purpose hardware/software environment for hard real time simulation of dynamical systems in the context of control engineering. With this system, realistic analysis and tests can be carried out for controller algorithms characterized by time determinism with strict restrictions, multiple inputs and multiple outputs and based on robust control theory. A hard real time system for plants simulation, controller implementation and testing (EPESC) has been developed, which is based on Real Time Linux (Linux-RTAI) and COMEDI project. The complete system provides a flexible and scalable environment for controller design, implementation and evaluation with hard real time restrictions. EPESC system has been developed by open-source software. The increasing prominence of open-source software is an important trend that has many positive benefits for scientific community.

Key-Words: Controller design, real time control, hardware-in-the-loop simulation.

1 Introduction

One of the main tools for testing and analysis of control systems is simulation, and if it must be carry out in a realistic environment for hardware and software testing before to develop prototypes, hardware/software-in-the-loop approaches are very useful and can support simulation results. Frequently there has been a gap between the control engineers who design the components of the control system and the software engineers who implement them. So that a thorough understanding of the methodologies used in both disciplines would make the overall process more cohesive. Component-based design of control systems has several advantages over the current practice of design, especially for stream-lining the transition from simulation to real implementation. Setting and abiding by standards for the interfaces between the components facilities rapid prototyping from simulation stage to the implementation stage.

The basic idea is: Ideally, controllers, signal processing/filtering components would not know if the information it is getting comes from the simulation code or from the sensor itself. Software components may be objects such as Java application or C/C++ program. Properties such as encapsulation, inheritance and polymorphism are useful for implementing components [1], [2], [17], [20]. Based in this approach, we have developed a new hardware/software environment for designing and testing control systems. Controllers are designed based on PID, H_2 and H_∞ methods, using an innovative procedure for controller auto-tuning based on experimental data and/or on theoretical mathematical models [10]. Model identification and controller design are carried out by means of a new software application for Windows family operating system: ControlAvH. This software makes transparent for the user the mathematical complexity associated with controller design, and additionally provides a friendly user interface which facilities monitoring, analysis and controller validation tasks.

In order to test control systems in realistic environments and to evaluate hardware/software in the loop performance, a hard real time system based on Real Time Linux under PC platform and on data acquisition cards has been developed: EPESC system. ControlAvH makes controllers tuning and by means of EPESC the controller is implemented and tested; for which ControlAvH-EPESC system provides a flexible and scalable environment for controller design, implementation and evaluation with hard real time tests. The complete system, EDECOS (Environment for Design and Evaluation of Control Systems), is used for testing and evaluating controller design by hard real time simulation.

The rest of paper is organized in sections as follows: In section two EDECOS is introduced, TuHiCo Toolbox is referenced, ControlAvH Tune application and EPESC system are described; in section three are analyzed the real time control system and simulation prerequisites, and in section four EPESC system requirements, basis architecture and functionality are defined, and finally, conclusions are summarized in section five.

2 EDECOS system

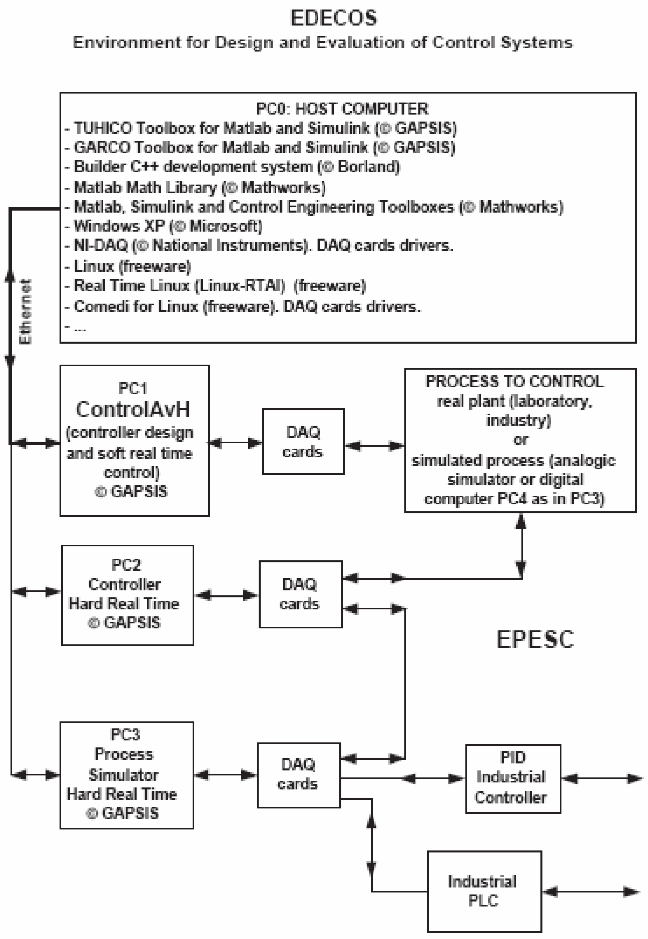
In our group GAPSIS, we have developed an Environment for Design and Evaluation of Control Systems (EDECOS), which consists of several components such as it is shown in Fig.1. The main parts of the EDECOS system are: 1) TuHiCo Toolbox, 2) ControlAvH application and 3) EPESC system. 1) TuHiCo Toolbox, based on Matlab, is used as previous phase to ControlAvH and its main functions are to develop efficient algorithms and methods for system identification, controller design and performance analysis. 2) ControlAvH application implements reliable methods and algorithms previously tested in TuHiCo Toolbox, and its main functions are data acquisition, system identification, controller computation, and user interface. 3) EPESC system provides hard real time plant simulation and hardware-in-the-loop environment for testing and evaluation of the designed controllers.

2.1 ControlAvH Tune Application

ControlAvH Tune has been developed with Builder C++ [3], which is a last generation RAD (Rapid Application Development) toll that incorporates a great quantity of standards with a very fast and efficient compiler. The base language is C++ which permits rapidity, flexibility and portability of the developed software. Builder C++ permits us to design the different interfaces that the user has to execute. In the main screen of the application the following elements are considered: 1) Graphical screen for signals evolution, 2) start and stop buttons of the control system, 3) on-line information of the run times of different tasks, 4) set of buttons to user the graphic system, 5) controller parameters for PID, H_2 and H_∞ and sample time.

By means of ControlAvH software application controller design and analysis are made. In order to carry out the application design we have used methods based on real time system techniques [1], [17], [20] and on automatic control theory applied to robust controller design [7], [9], [18]; with the objective to implement different phases related with controller design and validation.

Each application component can be represented with classes, objects and tasks within of the application. By means of object oriented programming (OOP) high flexibility is obtained, which permits an easy joint among different components. Besides, those facilities to develop structured software can provide the division of complex algorithms in smaller modules of easy resolution. Each one of components can be assembled within a generic part that can be implemented by an object oriented framework [1], [17], [20].


EDECOS
Environment for Design and Evaluation of Control Systems
PC0: HOST COMPUTER
- TUHICO Toolbox for Matlab and Simulink (© GAPSIS)
- GARCO Toolbox for Matlab and Simulink (© GAPSIS)
- Builder C++ development system (© Borland)
- Matlab Math Library (© Mathworks)
- Matlab, Simulink and Control Engineering Toolboxes (© Mathworks)
- Windows XP (© Microsoft)
- NI-DAQ (© National Instruments). DAQ cards drivers.
- Linux (freeware)
- Real Time Linux (Linux-RTAI) (freeware)
- Comedi for Linux (freeware). DAQ cards drivers.
- ...
Ethernet
PC1 ControlAvH (controller design and soft real time control) © GAPSIS
DAQ cards
PROCESS TO CONTROL real plant (laboratory, industry) or simulated process (analogic simulator or digital computer PC4 as in PC3)
PC2 Controller Hard Real Time © GAPSIS
DAQ cards
EPESC
PC3 Process Simulator Hard Real Time © GAPSIS
DAQ cards
PID Industrial Controller
Industrial PLC
Fig.1: EDECOS components and connectivity.

As it can be seen in Fig.1, the application has an operation mode (MODE 1) which works in soft real time controlling directly the process by means of data acquisition cards. Nevertheless, in MODE 2 ControlAvH works only as that in this case the controller (hard real time) is implemented in EPESC. Connection between ControlAvH Tune and EPESC is made by Ethernet. In MODE 1, ControlAvH works as autonomous system and it incorporates functions for controller design, analysis and controller implementation: which supposes an element of additional complexity.

Data acquisition and control algorithms execution for industrial processes require strict time restrictions and reliability, due to what the following functionality characteristics must be incorporated: 1) Coordination among real-time tasks, 2) processing of interruptions and messages within the system, 3) input/output device drivers to insure that they do not lose data, 4) inlet and outlet time restriction specifications of the system, 5) databases precision. To develop this software application with such restrictions, it is very important to take into account software engineering principles to avoid possible bottlenecks and to be able to get the optimal performance from hardware and software. ControlAvH has been developed under Windows XP, so that only soft real time is guaranteed.

Data from real time signals that the control application must register are the following: Time, process variable (PV), controller output (CO) and set point (SP), since error signal is obtained from $E = SP - PV$. These data are used by algorithms related with system identification methods, controller design and implementation, control system analysis, system simulation, as well as system temporary evolution visualization interface.

Commercial software tools for control systems, such as VisSim from Visual Solutions [21] or Matlab/Simulink from The MathWorks [13], are aimed at design and analysis. Matlab is used in conjunction with the Simulink package to provide a block-diagram-based graphical user interface along with some expanded simulation capabilities. To ease system implementation, the Real-Time Workshop Toolbox can be used to generate C code automatically from Simulink block diagram. VisSim works in similar way. Both products tend to generate monolithic code rather than component-base code. Matlab/Simulink has been mainly used for developing and implementing TuHiCo Toolbox (Tuning of H-infinity and H-2 Controller Toolbox) and GARCO (Genetic Algorithms for Robust Controller Design) Toolbox (see Fig.1). TuHiCo Toolbox is used previously to ControlAvH, and its function is to get reliable algorithms and methods; which, once have been tested and evaluated, are implemented in ControlAvH.

2.2 EPESC System overview

EPESC: Hard Real Time Control and Simulation system has been developed for PC-based controllers and PC-based plant simulators; due to PC-based environments are cheaper then industrial-grade processors and have a more open architecture. This open architecture means that third-party vendor is able to supply more of the components. Communication between PCs is based on the Ethernet hardware, as it is outlined in Fig.1. Low-cost communication suggested the user of TCP/IP or UDP/IP, which are nonproprietary communication protocols. The TCP/IP protocols guarantees, via implicit acknowledgment, receipt of data packets, but occupies a wider network bandwidth. The UDP/IP protocol is faster, but does not guarantee absence of packet losses. Basically, the communication between PC1 and PC2 consists of controller matrices, tuning parameters and data for controller analysis and fine tuning. For that, we have adopted the TCP/IP protocol.

The essence of real-time systems is that they are able to respond to external stimuli within a certain predictable period of time. Building real time computing systems is challenging due to requirements for reliability and efficient, as well as for predictability in the interaction among components. Real-time operating systems (RTOS) such as VxWorks [22], QNX [15] and LynxOS [8] facilitate real-time behavior by scheduling processes to meet the timing constraints imposed by the application. Control systems are among the most demanding of real-time applications. There are constraints on the allowable time delays in the feedback loop (due to latency and jitter in computation and in communication), as well as the speed of response to an external input such as changing environmental conditions or detected faulted conditions. If the timing constraints are not met, the system may become unstable.

EPESC system consists of hardware (input/output interface and electronic card for data acquisition) and a software application developed with C/C++ language, Linux Operating System and RTAI (Real Time Application Interface for Linux), [12], [16], [19]. RTAI lets to develop applications with strict timing constraints, but has the difference with respect to other real time operating systems (QNX, VxWorks, and LynxOS) that, like Linux itself, this software is a community effort and freeware. RTAI supports several architectures, such as X86/Pentium or PowerPC [16]. Such as it is shown in Fig.1, EPESC is used for hard real time controller implementation and for process simulator, both implemented with PC.

3 Real-Time Control and Simulation

3.1 Requirements

The real-time control and dynamic simulation software should:

- Satisfy the constraints of periodic real-time execution;
- Be able to interface itself with external processes, possibly with hardware;

- Be easily derived from models developed for off-line simulation.
- Additionally, must use the maximum open-source software packages.

3.2 Real-time execution platform

The purpose of obtaining a real-time application imposes the choice of an operating system capable of supporting the execution of real-time processes, [4], [17].

The Linux operating system extended with the Real Time Application Interface (RTAI) [12], [16] has been chosen. This operating System supports the execution of real-time processes, it is open-source, and it is widely used among the scientific community and in the European research centers.

3.3 Interoperability issues

Many custom libraries are available for Linux/RTAI. For the purposes of this project, the COMEDI (Control and Measurements Device Interface) [5] library, developed by the open-source community, is particularly interesting. By means of a set of standard interfaces, COMEDI allows managing the communication with hardware boards, and so provides a valid support for the data exchange on hardware channels. These drivers allow the access to data acquisition boards for analog and digital signals.

3.4 The software control application

A real-time software control system has been developed which emulates several functionalities of industrial controllers [6], [11]. This control application can be easily adapted to interact with simulations of industrial plant. Thanks to the adoption of COMEDI drivers, the control system can control without distinction a physical system, or a model based simulation of the system itself, provided that the two systems have the same number of input and output channels, disposed in the same order, [5], [17].

The software control system can be coupled with the real-time simulator, and each one of these two applications can be used as test bench when adding new features to the other one. So, the software control system can be used to test new and more refined industrial plant models, and to analyze their behavior, if compared to the behavior of the corresponding real plant, while the real-time simulator can be used to test some innovating control solutions, without taking the risk of damaging the plant hardware.

The Linux/RTAI operating system has been chosen for the control application too, for the same reasons explained in section 3.2. The controller application can execute a standard control cycle, with signals exchanged in Real-Time with the controlled system.

3.5 The closed-loop data acquisition

Both the control and the simulation applications should be able to transmit and to acquire signals on a hardware communication channel. In order to make any application unaware of the presence of hardware or software on the other side of the control loop it has been decided to implement COMEDI drivers for communications boards. The COMEDI package has been chosen because it is an open-source product widely used in the field of automation. Indeed COMEDI provides a standard for drivers of DAQ (Digital Acquisition boards) under Linux, [5].

A COMEDI driver for two National Instrument (NI) PCI-6014 boards has been used. Both boards are accessible from real-time processes: The first one is used by the controller process, while the second one is used by the plant simulator process, [14].

4 EPESC system platform

4.1 Architecture overviews

In the Fig. 2 EPESC architecture in case of an embedded digital controller is shown. The controller is embedded on available hardware at laboratory, such as: DSP, Field Programmable Logic Array (FPLA), card prototype and others. The plant simulator contains the following elements:

Plant simulator: The PC-based platform with Linux/RTAI operating system is used. Plant, sensors and actuators are simulated using their respective non-linear mathematical models, [13], [21].

A/D and D/A converters: the data acquisition hardware and the COMEDI project are used for input/output analog channels.

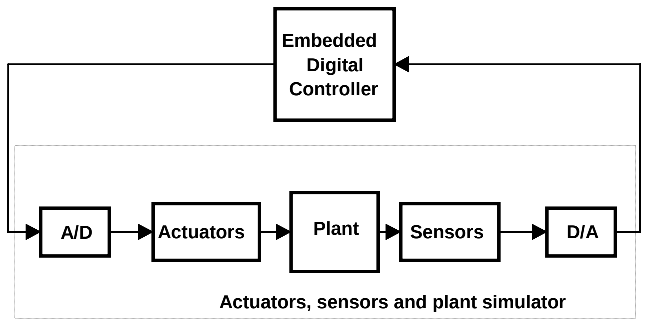

Embedded Digital Controller
A/D
Actuators
Plant
Sensors
D/A
Actuators, sensors and plant simulator

Fig. 2: EPESC – Embedded Digital Controller

In the Fig. 3 the EPESC hardware architecture is shown, where a digital controller is implemented on a

PC-based system and under Linux/RTAI. The same configuration, PC-based, is used for plant simulator.

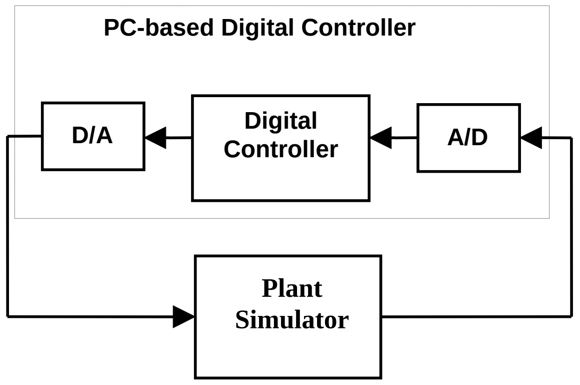


Fig. 3: EPESC - PC based controller

4.2 Modules and functions

In the Fig. 4 the EPESC software architecture block diagram is shown. There are two main components in this software architecture: The real time and the non-real time space components, both spaces are communicated by means of real-time first input – first output queue memory (FIFO). The Real time EPESC software is executed under Linux/RTAI like Linux module (in kernel space), the non-Real time software is executed under Linux like a process (in user space),

The real-time modules are implemented by means of three RT-threads, the so-called MONITOR (see Fig. 4) takes care of communication by FIFO with the system EDECOS and, concretely, with ControlAvH application. With this approach, the simulated plant can be modified if it was necessary; and the computed controller algorithm parameters are loaded. The state of the EPESC system is modified by command mode request and is controlled by RT-thread MONITOR. The RT-thread CONTROLLER or PLANT receive commands and parameters from MONITOR by means of global data blocks.

In order to reach the central idea of the EPESC system to evaluate the control systems, taking into account the hardware in the loop, input and output electrical signals are present, such as: The CONTROLLER signals (input process variables, PV, controller output, CO), and the PLANT signals (Output process variables, PV, manipulate variables, MV) are both wired signal interconnecting by means of two multi I/O data acquisition cards (see wired connections legend in Fig. 4). These cards provide the electric input-output signals among them.

All this system information must be sent to the Linux process so-called DISPLAY, which stores information on appropriate data structures and puts them in shared variables, available to the net services. The net services are implemented by two Linux thread (EPESC-SVR and EPESC-DL).

In the EDECOS environment context, the EPESC main server (EPESC-SVR) and the EPESC data-logging server (EPESC-DL) are used for ControlAvH Tune application to link with EPESC system; and request it services, which is achieved by means of command traffic.

The principal commands of EPESC-SVR processes are the following:

1) INIT: Set the system on the initial states; stabilize the plant and controller algorithm, and other initial actions.
2) STOP: Put the system in stop: Freeze the state.
3) MODE_1: In this mode, EPESC is used by ControlAvH like plant simulator (see Fig. 2).
4) MODE_2: In this mode, EPESC is used for plant simulation and digital controller test (RT-thread PLANT and CONTROLLER), for which are needed the controller parameters and the set-point.
5) SET_CONTROLLER: When ControlAvH has designed a new controller, via this command, send to EPESC new controller parameters in floating point double precision format (Sample time and values and dimensions of the system matrices A, B, C, D, in the space state model).
6) SETPOINT: When MODE_2 is present, the application ControlAvH sends to EPESC changes in the set-point. The digital controller computes the error signal (SP – PV).

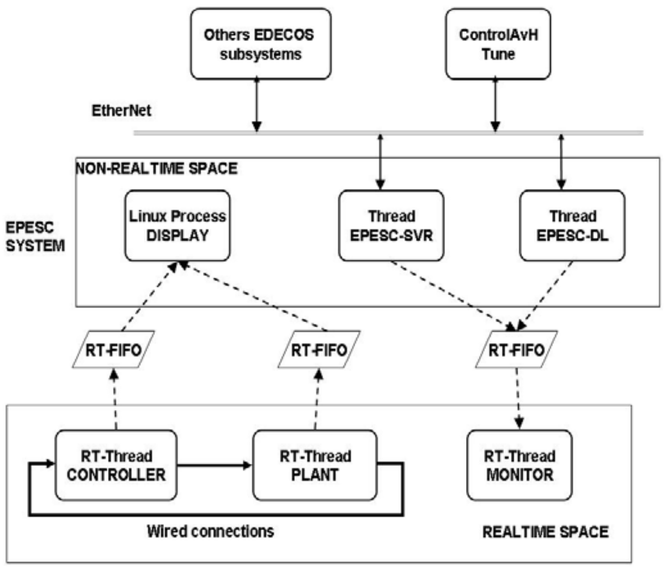


Fig. 4: EPESC - Software architecture.

In the other hand, the principal commands of EPESC-DL processes are the following:

1) REQ_DL1: When ControlAvH requests DL1 command, via data-logging, information about process variables is sent from PLANT to ControlAvH via EPESC-DL services.
2) REQ_DL2: When ControlAvH requests DL2 command, the controller output variables (CO) are provided by the ControlAvH application (usually in mode 1) via data-logging.

Eventually, in order to accomplish tests during the designing and implementation phases of EPESC system, and for making independent evaluations and tests over EPESC, we have made a client application that emulates EDECOS interface, and mainly, the communication with ControlAvH Tune application.

5 Conclusions

With EDECOS system we obtain a flexible hardware/software environment for hard real time plant simulation, controller implementation and closed loop control system evaluation.. On-line PID, H_2 and H_∞ controllers design and evaluation tasks are realizable by means of ControlAvH Tune software.

EPESC system has been developed by open-source software. The increasing prominence of open-source software is an important trend that has many positive benefits for scientific community. Open-source software is typically more stable and less buggy than proprietary software.

We have given specifications and technical solutions for EPESC system; keeping in mind, also, simplicity and economy criteria.

References:

[1] A. Abran and J.W. Moore, executive editors, *Guide* to *the* Software *Engineering Body of Knowledge*. IEEE Computer Society, 2004.

[2] B.S. Heck, L.M. Wills and G.J. Vachtsevamos, *Software Technology for Implementing Reusable, Distributed Control System*. IEEE Control System Magazine, pp. 21-35, February 2003.

[3] Builder C++. http://www.borland.com.

[4] Burns A., A. Wellings, *Sistemas de Tiempo* Real *y Lenguajes de Programación*, Addison Wesley (2003).

[5] COMEDI –The Linux Control and Measurement Device Interface. http://www.comedi.org/

[6] K.J. Astrom and B. Wittenmark, *Adaptive Control.* Addison-Wesley, 1995.

[7] K. Zhou, J.C. Doyle and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.

[8] LynxOS RTOS. http://www.lynuxworks.com.

[9] M.J. Grimble, *Robust Industrial Control: Optimal Design* Approach *for Polynomial Systems*. Prentice Hall, 1994.

[10] M.J. López, *Contributions to Advanced Process Control. Methods for H_∞ Controller Tuning*. University of Cádiz, 1999.

[11] Maciejowski, J.M., *Multivariable Feedback Design*, Adison Wesley, (1989).

[12] Mantegazza, y S. Papacharalambous, *"Real time application interface"*, Linux Journal (2000).

[13] MathWorks Inc. *Matlab*, www.mathworks.com.

[14] National Instruments Corporation. Measurement *and automation catalog*. http://www.ni.com.

[15] QNX Software Systems. http://www.qnx.com.

[16] RTAI. http://www.aero.polimi.it/rtai.

[17] S. Benne*t*, *Real Time Computer Control*. Prentice-Hall, 1998.

[18] S. Skogestad, I. Postlethwaite, *Multivariable* Feedback *Control*. Wiley, 2003.

[19] SuSE Linux 9.0, *Administration and user guide*, SuSE Linux AG (2003).

[20] V. Gazi, M.L. Moore, K.M. Passino, W.P. Shackleford, F.M. Proctor, J.S. Albus, The RCS Handbook. *Tools for Real-Time Control Systems Software Development*. Wiley, 2001.

[21] Visual Solutions Inc. *VisSim (Visual Simulator)*. http://www.vissim.com.

[22] VxWorks RTOS. http://www.windriver.com.