Distributed Software Platform for Automation and Control of General Anaesthesia

Gesner Passos, Nuno Roma, B. Andrade da Costa, Leonel Sousa, J. M. Lemos
IST/INESC-ID Rua Alves Redol, 9 1000-029 Lisboa PORTUGAL
Email: gesnerjr@sips.inesc-id.pt

Abstract

A parallel computer architecture and a distributed software platform for automation and control of general anesthesia is proposed in this paper. The system is a prototype research platform, intended to help on the development, simulation and test of new control algorithms for general anesthesia. It must be safe when used in real tests and flexible enough to allow the integration of new software modules. The system is composed by two computers, with the specific tasks of anesthesia control and process supervision. The platform makes use of TANGO, a specialized framework for distributed control systems, which provides software mechanisms useful to fulfill the project requirements. The architecture and the set of mechanisms proposed in this paper provide a high degree of flexibility to research on control algorithms, while ensuring the safeness of the whole procedure.

Index Terms

Distributed control, Control Systems, Automation, Fault diagnosis, Biomedical equipment.

1. Introduction

This paper describes the development of an automated anesthesia system, characterized by soft real-time and fault tolerant requirements. To fulfill these requisites, a parallel computer architecture and a distributed software platform are proposed.

The system is composed by two cooperative computers, inter-connected by a local network. One computer is responsible for the anesthesia control task. It executes the whole control loop periodically, which comprises the following procedures: receiving physiological data from sensors, executing control algorithms and commanding the actuators, to update the infusion rate of drugs. The main function of the other computer is to supervise and configure the control task operation. In order to provide the required safeness level, the platform has a set of mechanisms to detect faults and to give the operator the necessary information and tools to provide him with the ability to change the controller or to stop the operation of the automated anesthesia system if necessary.

The implemented machine is intended to be used at control system laboratories, to help on the development and simulation of novel algorithms, as well as in veterinarian clinics and hospitals, for *in vivo* testings. Any of these places adopt different equipment for physiological sensors and actuators. The anesthesia control for human and animals may be different in input and output variables. Therefore, the software platform shall be adaptable and configurable, specially in the sense of allowing changes in the interconnection between the several software modules.

As a research platform, the system must also be flexible enough to accept modifications on the control algorithms, without compromising its reliability. These features were accomplished by supporting the platform on the TANGO control system framework [1]. TANGO's structure provides distributed software mechanisms, tools and services, that facilitate the development of parallel systems. These mechanisms allow the interaction among the several software modules running in different computers and an accurate supervision of their activities.

Automation of the general anesthesia procedure has been an issue of great interest among researches. Nevertheless, there is still no full automated anesthesia system available. There are, at least, three projects in advanced stage to achieve this objective: the Target Control Infusion, consisting of a semi-automated control machine, but which lacks a reliable feedback signal to be considered a closed-loop system [2]; the McSleepy, developed at McGill university, which is claimed to enter the market within five years [3]; and a project at the Institut für Automatik of the Swiss Federal Institute of Technology Zurich whose platform has been under development for the last 10 years [4]. All these systems have a very specialized architecture and depend on a proprietary software to describe the control algorithm.

In contrast, the platform that is proposed in this paper exploits novel strategies of scalable, maintainable and reconfigurable control schemes, implemented through a distributed computational system [5] in order to support fault tolerant mechanisms and to allow the development of new control algorithms in a variety of software languages.

The paper is organized as follows. Section 2 presents an overview of anesthesia automation and the platform requirements. Section 3 describes the architecture of the distributed system and introduces the TANGO framework. A detailed description of the entire system and how the framework

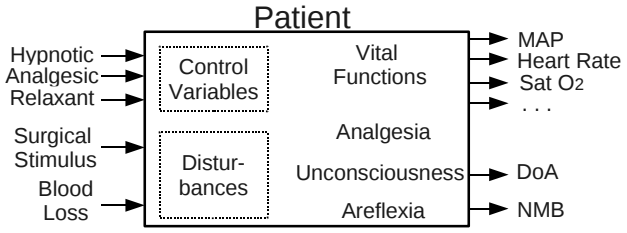

Figure 1. The patient model for anesthesia.

tools are explored to implement the system functionalities comprises section 4. The implemented prototype is presented in section 5. Finally, section 6 draws the main conclusions.

2. Automation of Anesthesia

To fully understand the system requirements of the computer platform for automation and control of general anesthesia, a brief overview of anesthesia will be provided. There are four main patient's states that are controlled and supervised by the anesthetist [4]: 1) *hypnosis* or unconsciousness level; 2) *analgesia*, concerned with the patient's pain sensation level; 3) *areflexia*, concerned with muscular activity and absence of movement; and 4) maintenance of vital functions (Figure1).

The anesthetist starts by applying several sensors on the patient's body. The Bispectral Index (BIS) is used to describe the Depth of Anesthesia (DoA) which refers to the level of consciousness/hypnosis. This index is directly related to the cerebral activity and is computed from the patient's encephalogram. There is no sensor for pain sensation, but the anesthetist is able to infer its level using indirect means, by considering the type of surgical act and by correlating the evolution of several physiological variables. The neuromuscular blockade (NMB) index is used to measure the muscle response to electrical stimulus. The maintenance of vital functions are described by a set of variables, such as the of mean arterial pressure (MAP), the heart rate, the saturation of O_2 in respiration, and the level of CO_2. All these variables are obtained with a physiological data acquisition system.

Anesthesia is induced by hypnotics, analgesics, and relaxant drugs, which are administered using automatic syringe pumps. It must be controlled during the whole surgical procedure, which may last several hours. As a consequence, the aim to develop the automatic anesthesia system is to contribute to the reduction of the anesthetist's workload in repetitive procedures and to allow him to be more focused on the patient's state.

The basis of the automatic anesthesia system is a complex nonlinear model that is used to describe the relationship between the administration of the anesthetic drugs and the effects that are induced in the patient's body. At the current research stage, two controllers, denoted as DoA and NMB, are used to control the depth of anesthesia and the level of neuromuscular blockade [6]–[8].

2.1. System Requirements

The proposed platform will be used to develop novel algorithms and to evaluate their performance, both in veterinarian clinics and hospitals. It has three characteristics that establish its main requirements: 1) it is related to people safeness; 2) it is a distributed platform; and 3) it should be expandable and reconfigurable, in order to allow the development of new control algorithms or the addition of new functionalities.

The main requisites are the following:

- *Reliable and stable*: The system should operate during the whole surgery process, which may last several hours, without interruption;
- *Modular, expansible and maintainable*: The platform software modules may change. There are two main reasons for this. First, new algorithms may be implemented and tested. Second, veterinarian clinics and human hospitals have different sensors and actuators machines. These basic environment adaptations and manipulations should be done with minimum error prone source code edition. Ideally, only static configuration data should be changed, in order to perform these adjustments.
- *Safety*: Although the patient's safety is assured by having all tests accompanied by a control engineer and an anesthetist, mechanisms to stop the automated control system shall be provided. Faults must be detected, and the operator shall be advised about all malfunctions.
- *Archiving data*: All data collected from the surgeries shall be available for future analysis and research on a reliable database;
- *User friendly*: The system must be visual appealing and intuitively enough to be used by both anesthetists and control engineers.
- *Remote assisted*: The software engineer team is often far away from hospitals, where *in vivo* testings are applied. They should be able to access the machine, check the logs, and correct errors.

3. Platform architecture

The architecture of the platform corresponds to a general closed-loop controller, as shown in Figure 2. It is composed of two computational units: the control and the supervisor units. The control unit interacts with a physiological data acquisition system to read its input signals, it executes the control algorithm to maintain the physiological variables as close as possible to the established references, and it handles the syringe pump actuators to change the drug infusion rates. All these tasks must be executed periodically. The supervisor unit monitors the control task, provides a visual

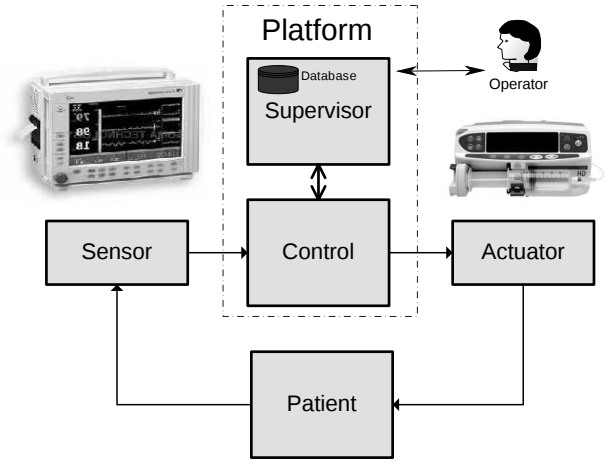

Figure 2. Platform architecture.

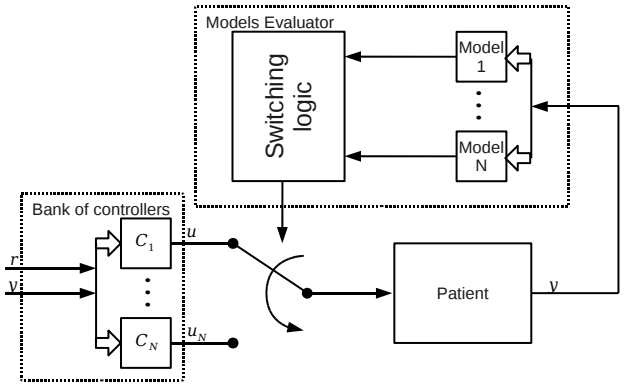

Figure 3. Model of an implemented DoA controller.

interface to the operator and deals with the system database. Two main reasons are behind the adoption of this parallel and distributed structure: the computation burden of the control algorithm and safeness, as described in the following sections.

3.1. Control Algorithm

The great challenge of the control of anesthesia is related to the great variability of the physiological parameters from one patient to another. Furthermore, the patient parameters may even change during the surgery. To tackle this particularity, both NMB and DoA controllers adopt a switched model adaptive control strategy [6], [7]. Figure 3 shows the complexity of one of such controllers under research. The controller uses a bank of models that should be simultaneously run in order to infer the best respective controller to apply. Currently, the NMB controller handles 100 analytical models, designed to cover all available data for general anesthesia of patients ASA[1] I to IV, where each model is defined by 8 parameters [7].

The control algorithm execution is not only computationally demanding but, while the research advances, it may become more complex as long as new inter-relation of physiological variables are considered and nonlinear strategies are applied. Hence, by allocating a whole computer for the control task it shall ensure that new algorithms may continue to evolve without exhausting the computational resources.

3.2. Supervision and safeness

The platform will be used for research and development of new control algorithms. These algorithms shall be written by control engineers, whose programming skills are not known *a priori*. Furthermore, the algorithms may be complex enough so their its execution may surpass the available time interval. These considerations demonstrate the requirement for software mechanisms that prevent and detect real-time malfunctions. In fact, three main issues must be evaluated and inspected in the controller: if it fulfills the time requirements, if the input physiological signals are within a safe range and if the amount of infused drug, as well as its rate, are also at an acceptable level. The supervisor shall monitor all these parameters. Hence, in order to ensure a fail-safe condition in the supervisor unit, despite the control operation task, the supervisor does not share the same computational resources of the controller. Instead, it runs in a different and independent unit, where the supervisor monitors the controller through a dedicated Ethernet link that is established between the two units. If necessary, the supervisor may stop the controller and change the system to manual operation.

With the presented approach, it is provided a wide level of flexibility and efficiency in the control algorithm execution, ensuring that it operates under safe constraints.

The supervisor still has two additional functions: it is responsible to provide a visual interface to interact with the operator, who may be either an anesthetist or a control engineer; and it manages the database, keeping a record of static configuration parameters, as well as dynamic data related to the anesthesia in a surgery operation.

3.3. Framework

In order to implement the required anesthesia control system , is was adopted a software framework to build the distributed platform. This framework provides the structure that establishes the inter-communication between the different software modules distributed in both computers and some other tools, thus making the system implementation easier and decreasing the development time. Unfortunately, this kind of framework is not available for medical systems.

1. American Association of Anesthesia score http://www.nda.ox.ac.uk/wfsa/html/u14/u1405_01.htm.

As consequence, an adaptation from another field of application was required. To be acceptable, the requirements of the adopted framework should be, at least, as strict as the current application. Among some of the most preeminent open-source control system frameworks (ACS, EPICS, TANGO) [9], two of them (EPICS and TANGO) were developed in the scope of particle accelerator control systems, as described below:

- The Experimental Physics and Industrial Control System (EPICS) [10], [11] was developed in 1989 at Los Alamos National Laboratory. It has successfully supported the construction, test, and integration phases of many particle accelerators. In addition, it has also been used for astronomy, water and electric distribution and several industrial applications. Its main idea is the definition of a distributed processing database, defined as a set of process variables that are available over the network through identificators corresponding to their names. These variables allow sequencialization, control IO operations, closed-loop algorithms execution, data collection, general logic, etc.
- The TACO Next Generation Object (TANGO) [1], [9], [12] has taken the best contributions of its predecessor Telescope and Accelerator Controlled with Objects (TACO), and added some missing new features. TANGO was first presented in 1999, at the ICALEPCS conference in Trieste [1]. Currently, there are, at least, 4 particle accelerator institutes working at the framework improvement.

The usual application environments of these frameworks have been very large distributed control system that often include multiple software layers, many types of graphical user interfaces, hundreds of distributed processors, and many different computer based devices [13]. These environments require a dynamic computer system that allows an easy inclusion of new monitors, consoles and other devices, as well as the removal of some of these components. Therefore, this kind of frameworks are good alternatives for the current platform.

The two considered frameworks provide some important features [9]: 1) inter-communication between distributed software modules; 2) easily built user interface, between the operator and the control system; 3) alarms management, to help the supervision task; 4) data management application, used to store in the database static and dynamic data from the control system; 5) central management of logs, to keep status information about the software modules.

Although both frameworks provide all these features, as well as many other similarities, TANGO was chosen for this platform. Its main features are discussed in section 4, but two main characteristics determined its choice. First, the system may be designed as a set of components, inheriting the benefits and design clearance from an object-oriented approach. Second, it defines a central repository database, which may be used by all modules to configure their behavior. This characteristic provides an easy configuration of the system, the addition of new algorithms and the exchange of components without re-edition of the platform source code, thus improving the system maintainability.

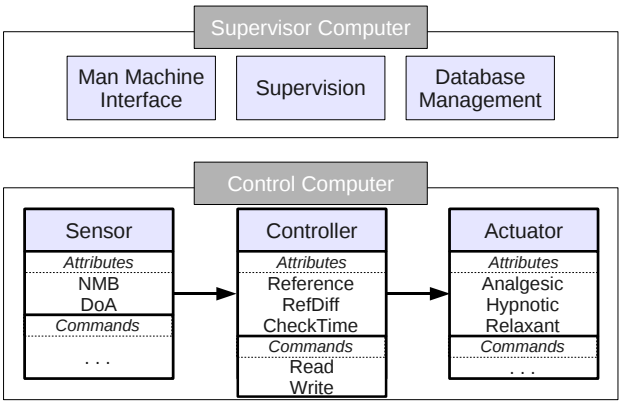

Figure 4. Software modules overview.

4. Distributed Software

This section details the software organization, illustrated in Figure 4. It describes the main components and their implementation through the framework toolkit. First, the modules interaction is discussed. Then, it is presented the implementation of the control and supervisor units.

4.1. Interaction between the parallel processes

The TANGO communication module is based on CORBA [14], Common ORB (Object Request Broker) Architecture. CORBA is a widely accepted standard, created and managed by the Object Management Group (OMG) to allow communication and interaction of object oriented systems through the network. It defines APIs, communication protocols and service information models to enable heterogeneous applications, written in possibly different languages and running in multiple platforms, to inter-operate. CORBA therefore provides platform and location transparency, for sharing well-defined objects across a distributed computing platform [9].

Nevertheless, TANGO framework hides the CORBA expert issues (such as communication protocol, server-client reconnection, etc.) from the software developer, making them transparent to the application. As a result, must of the times, it is only necessary to know the name of the device (see section 4.1.1) to interact with it, while the device may be local, at the same address space, or running in a remote machine.

4.1.1. TANGO's Device Concept. One of the TANGO strengths relies on the definition of a single object, denoted as device, from which all others derive. Device, as the name suggests, may be used to represent system components, such as system peripheric devices (hardware), or even to denote software components.

Each device is hosted in a server process, which allows it to accept commands from many different clients. Serialization of requests, creation and destruction of threads and timeout protections are managed by the framework and are transparent to the software developer.

Each device is characterized by four items:

- **Properties**: corresponding to the configuration parameters of the device (e.g. baud rate of a serial device). They are kept in the system database and are re-loaded at the device initialization.
- **Commands**: the actions offered by the device, similar to methods in C++ objects. TANGO commands have a fixed structure: command name, single input data and single output data. Although fixed, the offered data types are flexible enough to implement all the commands required by the implemented system.
- **Attributes**: special characteristics of data elements . There are only two operations allowed for attributes: *read* and *write*. They naturally fit the physiological signals that are considered for this application (e.g. NMB, DoA, drug infusion rate, etc). Attributes have some important properties and behaviors. For each attribute, it is possible to define the label, the measurement unit, the acceptable range, etc. Furthermore, these ranges are evaluated for every *read*/*write* operation of the attributes, automatically triggering an alarm if the current value is out of range.
- **States**: Special attributes, available in all devices, that allow them to implement up to 15 modes of operation. Each state may define a set of allowable commands and operations on the attributes. This prevents the clients from manipulating the device outside the definition that is specified by that mode of operation. For instance, an actuator device in the OFF state does not allow the *write* attribute operation, i.e., it does not allow drug infusion.

4.1.2. TANGO's Database. The database device is one of the principal components of TANGO framework, supported on a MySQL database. The main purpose of this database is the maintenance and storage of the records related to the CORBA framework that are required to connect to the several devices, making the interaction and identification of devices in a network possible through their names. Another functionality of the database is to store the configuration data related to devices and their attributes. These parameters may be changed off-line. This feature is exploited to provide the flexibility of the environment, allowing the system reconfiguration and modification.

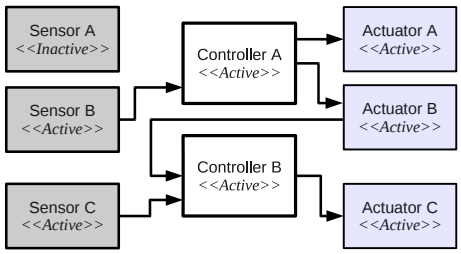

Figure 5. Software components inter-connection of the controller unit.

4.2. Controller Unit

In Figure 5 it is illustrade the software components interconnection of the controller unit. The system is said to be reconfigurable, because it can be used in different surgery rooms and provide support for different anesthesia control algorithms. It may have many different sensor and actuator devices, one for each different machine in a surgery room. For example, automatic syringe pumps are used to perform intravenous infusion of drugs in human patients, while in rats, anesthesia drugs are delivered through vaporizers. For each environment, the operator may select which devices should be activated.

The controller devices establish the interconnection between the different modules. As illustrated in Figure 5, a controller may request any available input attribute valuen from any sensor device or even from an actuator device (e.g. actuator B in Figure 5). These connections are defined by configuring the properties of the device.

The configuration of these interconnections are performed through the database manipulation applications provided by the adopted framework, and do not require changes to the source code. This makes the system highly maintainable and ensures consistency throughout the inevitable changes that must be done during the machine's lifetime [13].

4.2.1. Sensor Interface Device: . The sensor interface device interacts with the physiological data acquisition machines, thus configuring, extracting the information and implementing the machine's specific protocols. From the point of view of the platform, each device must implement two functionalities. Firstly, the available input signals should be implemented as readable attributes (For example, the NMB value shall be obtained from the sensor through an attribute *read* operation). Secondly, it should ensure the synchronization.

This second requirement deserves further explanation. The adopted framework supports a publish/subscribe communication paradigm, that allows data from a single device to be received by multiple clients, providing a more efficient use of the network bandwidth and allowing an easier synchronization procedure [13]. With this event notification

service, clients may subscribe themselves to the events that are related to the attributes of any device. This service ensures that, whenever a device's attribute changes, the new attribute value is sent to all subscribed clients. Furthermore, considering that states are treated as special attributes, clients may also subscribe themselves to listen to the devices state changes.

Hence, each sensor interface device shall generate attribute events every time it receives a new physiological value. These events may be a source of synchronization that will allow the control unit to execute a further step in the control algorithm.

4.2.2. Actuator Interface Device: . The actuator interface device interacts with the actuator machine (syringe pump), which is responsible for the administration of the anesthesia drug. It must control this machine, configuring it and implementing the communication protocol. The main requirement for this device is the implementation of the *write* and *read* attributes. Every time a *write* attribute request is received, it must act over the actuator machine to change its operation level (e.g., the infusion flow rate).

4.2.3. Controller Interface Device: . The controller interface device is the very responsible for the flexibility and safety of the system. Nevertheless, this device does not execute any control algorithm on its own. It encapsulates the important and relevant input information and provides an interface to the client process that actually implements the algorithm. The controller interface has two commands: *read* and *write*. The client program executes the *read* command to receive the input data (sensors values, and references), and the *write* command to send the controller the new values to be sent to the actuators. With this procedure, the *read* command establishes the synchronization of each control step.

Defined like this, the only requirement of a given control algorithm process is that it should perform this two commands over the corresponding controller interface device. All others issues, such as programming language, mathematical libraries, even the machine where it will be running, are not constrained, since TANGO allows the client procedures to run on remote machines, as long as they can properly define the device name.

Each controller process is defined by three types of attributes: input synchronization attribute, array of input attributes and array of output attributes. The controller subscribes itself to listen to the changes of the attribute value of the input synchronization attribute. Whenever it receives a *read* request, it first waits the reception of this attribute. Then, it performs the *read* operation on all elements of its array of input attributes to refresh their values. Finally, it sends the client all inputs values from the system. Whenever it receives a *write* request, it does the *write* operation on all elements of the output array.

The controllers also have two additional attributes to help on the performance evaluation, discussed in section 3.2: *RefDiff* and *CheckTime*. The first measures the difference between the control reference and the current physiological value. *CheckTime* is a measure of the execution time of the last step of the control algorithm. These attributes serve for two purposes. Firstly, to verify that the control algorithm fulfills the real-time requirements, by ensuring that *CheckTime* is always close to the sampling period defined for the control algorithm. Secondly, for on-line supervision of the stability of the closed-loop control system, by inspecting the growth of the *RefDiff* attribute.

4.3. Supervisor Unit

The Supervisor unit has four main purposes: 1) configuration of the control unit environment; 2) supervision of the control algorithm execution; it must detect faults and provide mechanisms to safely stop the system or to change the considered controller; 3) archive of the available data related to anesthesia; 4) establishment of an environment to interact with the system operator.

4.3.1. Supervision:. The operations conducted by the supervision module are: 1) detection of faults; 2) notification of the operator for their occurrence; and 3) provision of mechanisms for reacting. This unit will identify two kinds of faults: internal faults occurred within any device running in the control unit, and faults in the control algorithm. To achieve such functionality, the supervision task relies on two services provided by the framework: the event notification and the alarm generation.

To identify any device malfunction, the supervisor subscribes itself for notification of changes in the state value of all devices. When any device recognizes an internal exception, it changes its state to the FAULT state. This event is received by the supervisor, that notifies the operator.

As it was described in section 4.1.1, the controller device has two attributes to allow the evaluation of its performance. By defining the allowed limits for these attributes, it ensures the monitorization of the input range and of the execution time. As an example, at every *read* and *write* operation the framework verifies if the value of the corresponding attribute is within an acceptable range. If it goes out of range, the device changes its state to ALARM. In order to be able to identify an unstable condition, it is necessary to define the allowed range for the sensor and actuator attributes. For example, the operator may define the maximum amount of drug that is allowed to be infused into a patient. If this value is exceeded, the actuator device goes into ALARM state and the operator is alerted for such occurrence.

In the event of the supervisor detects any malfunction in the control algorithm execution, it also notifies the operator. The operator may then send a command to stop the controller device, preventing it to answer the invalid *read* and *write* command requests, thus stopping the automatic control. In such situation, the operator may activate a backup controller, restart the automatic control, or change to manual operation.

4.3.2. Data Management:. At any instant, the operator may choose the several attributes that shall be archived for each anesthesia procedure. For example, it can configure the system to periodically sample and store the NMB and BIS values, their respective references, as well as the analgesic, hypnotic and relaxant drugs infusion rates. These values may then be exported to different application formats, to support the research and the evaluation of the automatic control operation.

4.3.3. User interaction:. The system provides two different environments. One is the configuration environment, used by the control engineer to upload new algorithms, configure the system, activate and deactivate devices, etc. The other environment is to be operated by the anesthetist. This mode hides most technical details of the system. The adopted framework provides some applications that support the design of intuitively and easy to use user interfaces.

4.4. Special features

One special requirement of this platform is the possibility to test and evaluate different control algorithms. This means that control engineers may develop and test their control algorithms without requiring special knowledge about the distributed software architecture. It is also desirable not to require any special knowledge on any specific programming language, allowing them to develop the algorithms with the tool they are more comfortable with (for example, they may use Matlab to describe their algorithms). Since the adopted framework has bindings for Matlab, LabView, C/C++, Java and Python, all these languages may be used to describe the control algorithm and to interact with the controller device, using the *read* command for periodical input values and the *write* command to refresh the outputs.

Another feature of the implemented system is the possibility to get information of the entire platform from a remote computer, by simply accessing the system database. This greatly facilitates the remote assistance, since the software engineer team does not depend on the operator information to be able to identify a fault and to propose corrections.

5. Prototype

Figure 6 illustrates the developed prototype. Each unit is composed by one VIA EPIA-LT Mini-ITX embedded board, equipped with a VIA C7 processor and with 1GB of RAM, running Linux operative system. Linux was chosen not only because it is license free, but also because it is easier to integrate with other modules, through the available device drivers.

The control unit interfaces with 4 devices through RS-232 serial ports. One is linked to the physiological data acquisition system, a Datex Ohmeda S/5 equipment. It is a modular monitoring device, that interacts with several sensors, allowing the visualization of several physiological parameters. In the adopted platform, it is implemented as the sensor interface device that provides the NMB, BIS and other physiological values. The other three ports are interconnected to three fully featured syringe pumps for general infusion application (Alarys GH Syringe Pump). They deliver three drugs through intravenous routes: an analgesic, a hypnotic and a relaxation drug.

Both computers are inter-connected with a 100 Mbps Ethernet link. An extra Ethernet connection is provided to connect the supervisor to another external computer or, eventually, to the Internet. Interaction with the user is done through a touchscreen handled by the supervisor. In the event of a supervisor failure, this screen may be connected to the control computer through a KVM switch, available at the platform, providing the ability to keep the system running with a direct interaction of the physician with a minimalistic console.

6. Conclusions

The proposed architecture provides the mechanisms for the development of a flexible but reliable distributed control system for the automation of anesthesia. The adopted distributed structure allows a specialization of each computational unit and provides an efficient execution of the control algorithm. The modularity of the platform architecture offers the ability to modify the system environment without interfering with the other modules.

The system is built around a central database repository, to allow the system reconfiguration and maintenance and to avoid significant source code changes, thus improving the system reliability.

The usage of the TANGO framework greatly reduced the development time and increased the reliability of the interoperation between the several distributed units and software modules. The facilities offered by this framework were extensively exploited to implement a supervision method of the each control algorithm under execution. Considering that this framework provides a scalable structure for the development of distributed systems like this, the implemented platform may be easily adapted to implement more complex control algorithms, being able to comply with the implied real-time requirements.

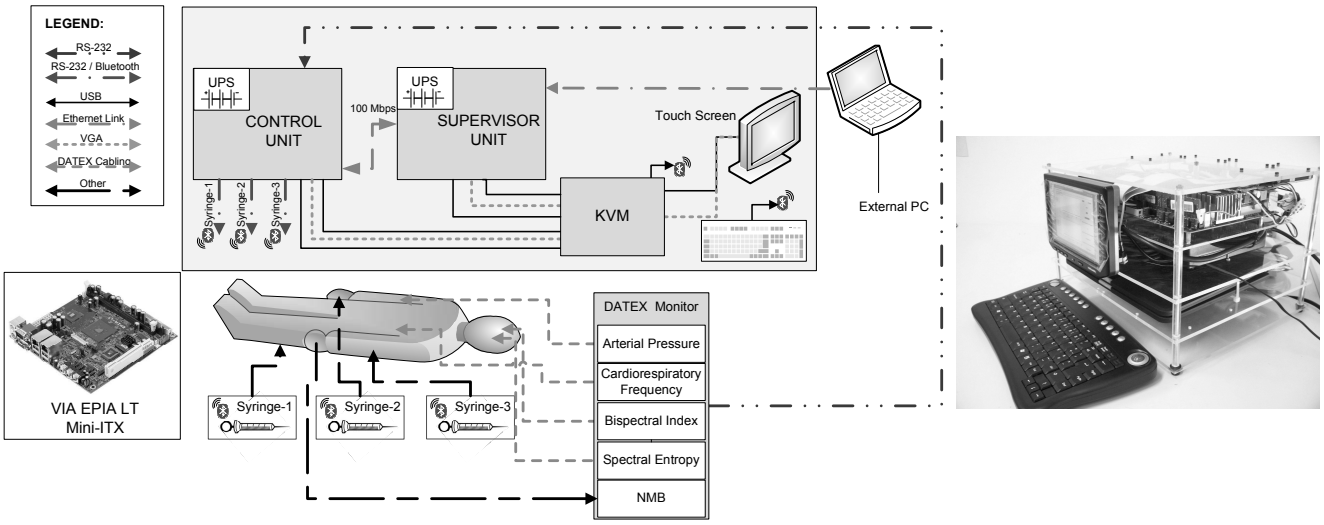

Figure 6. Overview of the developed platform for automation and control of anesthesia.

Acknowledgments

This work has been done under the framework of project IDeA - Integrated Design for Automation of Anaesthesia, contract PTDC/EEA-ACR/69288/2006.

References

[1] A. Götz, JL. Pons, E. Taurel, and P. Vervier, "The TANGO control system," *ICFA Beam Dynamics Newsletter*, vol. 47, December 2008.

[2] C. H. Ting, R. H. Arnott, D. A. Linkens, and A. Angel, "Migrating from target-controlled infusion to closed-loop control in general anaesthesia." *Computer Methods and Programs in Biomedicine*, vol. 75, no. 2, pp. 127–139, 2004.

[3] TM. Hemmerling. (2009, March) Developing a closed-loop control method for an automated anesthesia system using NI LabView software. [Online]. Available: http://sine.ni.com/cs/app/doc/p/id/cs-11762

[4] C. Frei, "Fault tolerant control concepts applied to anesthesia," Ph.D. dissertation, Automatic Control Laboratory - Swiss Federal Institute of Technology ETH, 2000.

[5] I. Song, F. Guedea-Elizalde, and F. Karray, "CONCORD: A control framework for distributed real-time systems," *IEEE Sensors J.*, vol. 7, no. 7, pp. 1078–1090, July 2007.

[6] N. Cardoso, "Predictive control of depth of anaesthesia," Master's thesis, Grenoble INP - ESISAR, 2008.

[7] J. M. Lemos, H. Magalhaes, T. Mendonca, and R. Dionisio, "Control of neuromuscular blockade in the presence of sensor faults," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 11, pp. 1902–1911, Nov. 2005.

[8] J. M. Lemos, N. Roma, T. Mendonça, L. Sousa, B. M. D. da Costa, C. Nunes, P. Amorim, and L. Antunes, "Developmentof an integrated control system for anaesthesia automation," *Controlo 2008, 8th Portuguese Conf. on Control Automation*, August 2008.

[9] B. Lopez, "Non-commercial frameworks for distributed control systems," *European gravitational Observatory*, January 2007.

[10] M. Clausen and L. R. Dalesio, "EPICS – experimental physics and industrial control system," *ICFA Beam Dynamics Newsletter*, vol. 47, December 2008.

[11] L. R. Dalesio, M. R. Kraimer, and A. J. Kozubal, "EPICS architecture," *Proceedings of International Conference on Accelerator and Large Experimental Physics Control Systems*, pp. 278–282, 1991.

[12] The TANGO team, *The TANGO control system manual*, Version 6.1, April 2008. [Online]. Available: http://ftp.esrf.fr/pub/cs/tango/tango_61.pdf

[13] K. S. White, "Status and future developments in large accelerator control systems," *Prepared for 9th International Computational Accelerator Physics Conference (ICAP 2006)*, Oct. 2006.

[14] C. McHale, *CORBA explained simply*, October 2004. [Online]. Available: http://www.ciaranmchale.com/corba-explained-simply/