Design of a distributed system architecture including an automatic code generator

Jose A. de Frutos\textsuperscript{a,*}, Jose M. Giron-Sierra\textsuperscript{b}

\textsuperscript{a}Department of Automática, Escuela Politécnica, Universidad de Alcalá, Alcalá de Henares, 28871 Madrid, Spain
\textsuperscript{b}Department of ACYA, Fac. Físicas, Universidad Complutense de Madrid, 28040 Madrid, Spain

Received 27 September 2001; revised 12 March 2002; accepted 18 March 2002

Abstract

Distributed monitoring and control systems play an important role in our time, with many new applications. The purpose of the research described in this article, is to propose a complete solution for the fast and easy implementation of that type of system. Both hardware and software aspects are covered. The hardware is based on a simple bus, using modem for the communications. The nodes are built around embedded PCs for powerful (smart) features. Circuits are provided for the direct use of standard sensors. Software tools have been developed for all the usual steps to get a running system. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Distributed systems; Automatic code generator; Fieldbus

1. Introduction

An important, contemporary trend in practical control systems is toward distributed solutions, instead of centralised solutions [1–5]. Many technical reasons support this trend, like scalability and noise immunity. Another relevant, strategic reason is that it is a powerful approach for complex problems: divide and conquer.

Communications is an important topic in distributed control systems. They must be reliable and adequate for the control objectives. There are several architecture alternatives [6]. Some of them are like conventional networked computer systems, adding specific adaptations for control applications. Others think that simpler is better. This is the origin of fieldbuses. They tend to implement only the bottom ISO network protocol layers, since this is enough for industrial plants and makes the communications be easier and faster [7,8].

Among the different technologies that can be used to build control nodes [9], embedded PCs offer very interesting characteristics. With embedded PCs, the software development for applications is easy. The computational power of PCs is an important advantage, which can be used for more performances of the nodes (for instance, intelligent features).

This article describes a system that implements the previous ideas. One of the objectives of the system is to make easier the building of distributed control applications by people with moderate technical knowledge.

2. Design principles

Both hardware and software aspects are considered in our work, in order to make easy the implementation of distributed control systems. Wiring will be simple, and programming will be done with a software tool for automatic code generation.

The structure of the system is based on a bus connection, which is supervised by a main PC computer. There is a bus control unit, to connect the main PC to the bus using the serial port of the PC: RS-232. The bus supports a communication through modem. Currently normal telephone line is being used for the bus, so everything related to cables, connectors and tools are standard and easy. Other alternatives of communication, such as wireless means, are also possible to use. Fig. 1 shows the block diagram of the system.

Each node has three different blocks: one for communications through the bus, other for interface with a process (using digital an analog I/O), and the last for processing and control functions. The process interface includes signal adaptation and conditioning to allow for the direct connection of standard sensors.

The generation of control software is made with a
be or not activated. If a node has been given the permission to learn, it starts to learn from the main PC during the operation of the system. The main PC takes some decisions, when certain events happen (for instance, an alarm from a node). When there are same decisions for same events at a node, several times, this node begins to take these decisions autonomously (the node sends information about it to the main PC; anyway the main PC can always override the node’s decisions).

An example of possible application could be a chemical plant with several units—tanks, reactors, etc.—separated by relatively long distances. The idea is to assign a node to each unit of the plant, for monitoring and/or control functions. Long distances are not a problem with our system: the nodes can be easily interconnected to a central control room. In this kind of application, the sampling times are not very fast and the timing requirements for communications are not stringent.

3. Communications

The communication system is like a fieldbus, with the possibility of a fast handling of alarms using an interruption mechanism. Modern transmission is used for noise immunity, long distance capability and easy implementation. The speed chosen for the data is 1200 bauds, enough for many industrial processes (however, other higher frequencies can be also decided, with no problem). A modular concept is observed for the system architecture, including the internal structure of nodes. In this way, it is easy to change the communication support: wire, radio or optical fibre.

Protocols are provided for routine periodic exchange of
Fig. 3. Circuit for interrupt detect.

Fig. 4. Node circuit for communication control.
information between nodes and main PC, and for event-driven asynchronous behavior of the system.

During routine operation, the main PC asks periodically to the nodes for information. When there is an event that the main PC must know as soon as possible, the node detecting the event places an interrupt. To do that, the node puts in the bus a signal of different frequency than the frequency used by normal data (for the case of 1200 bauds for data transmission, 22 KHz are employed for the interrupt signal).

Figs. 2–5 show the circuits used for the communication block in each node.

4. The node

The heart of each node is an embedded PC. It consists of a Flashlite board by JKmicrosystems. It is a small, compact board (10.7 cm x 9.2 cm) with a V25 NEC microprocessor, 100% Intel compatible. The board has 512 Kbytes of pseudo static RAM and 256 Kbytes of flash memory [10]. It is easy to develop programs for this board: a standard PC can be used for editing and compiling the executable code, and then this code is downloaded to the board for real application.

A program has been developed to give the desired functionality to the node. Adhering to the object oriented methodology [11,12], this program is made with a set of objects according to the taxonomy described in Fig. 6. There are two main layers in this taxonomy. The top layer includes a main object, 'Node', which handles two interface flows: one is the exchange of messages with the bus, and the other is the interaction with the external process. The 'Ctr_A' object responds to analog signals, using the 'PID' object for PID control and the 'Transfer' object for a more flexible control (configured by the user, giving a description in terms of controller discrete transfer function). The 'Ctr_D' object responds to discrete on-off signals. The 'Alarms' object exerts actions in response to alarms. The bottom layer of the taxonomy includes the objects to handle the A/D and D/A converters, and the discrete on-off I/O lines.

The embedded PC interacts with the monitored/controlled process using digital and analog interface circuits. An eight channel 12 bit A/D converter (the MAX 186 by MAXIM) is used for analog measurements. By means of circuits for signal conditioning and level adaptation, many standard sensors can be connected to the node. Actually, there are four kinds of inputs to the node: PT100 (for temperature measurement), pH sensors (mV level), 4–20 mA loop, and signals inside a voltage range (a gain-controlled amplifier is used for this, so voltage ranging

Fig. 5. Interrupt signal generator circuit.

Fig. 6. Object-oriented taxonomy of the software in the node.
between 0.0.1 and 0.40 volts can be handled). Fig. 7 shows the circuits for analog inputs.

By means of a dual 12 bit D/A converter (the MAX 532 by MAXIM), the node can generate two analog outputs. In addition, the node has six digital inputs and six digital outputs.

The node can use any selected analog input and output to work as a PID controller or in a more general way, as a controller implementing a discrete transfer function. Also, any selected digital inputs and outputs can be used by the node to work as a PLC (programmable logic controller). Other analog or digital I/O channels of the node can be used by the main PC (for instance for tele-measurement purposes).

During configuration of the nodes, the main PC is used to specify the functions of each node (controller and/or

---

Fig. 7. Node circuits for signal conditioning.

---

Fig. 8. Orders for communications in the system.
measurement unit). Part of the specification is to assign analog and digital channels to the functions. Also, the tuning of the PID, the numerator and denominator of the transfer function, the behaviour of the PLC, and the level conditioning of the analog signals.

5. Communications protocol

The main PC and the nodes can exchange messages using a simple protocol and a repertory of possible orders. For the normal operation of the distributed control system, a polling strategy is followed: the main PC asks sequentially, one node after one, for information and answers to each node with the pertinent orders.

A compact language of orders has been designed, according to the capabilities of the nodes. Fig. 8 shows a summary of the language. Since the repertory of orders is limited to few alternatives, the orders can be encoded with single ASCII characters. In this way, the exchange of messages between nodes and the main PC can be fast.

6. Code generator

The implementation of a distributed monitoring/control system with our hardware/software solution involves three main steps. The first is devoted to hardware: nodes, sensors and actuators, wiring; interconnection of nodes and the main PC through the bus. The second is to configure the functions of the nodes. The final step is to create the main application, so the main PC can govern the distributed system, and accomplish the measurement and monitoring functions.

Several software tools have been developed to make easy the three steps of the system deployment. Some of the tools are made for testing purposes during hardware implementation and system configuration. Another tool, that can be considered the most important, is an application development environment with automatic code generation capability [13,14]. With this environment the process of creating an application is a matter of choosing human interface components, and assigning functions to these components [15]. Several types of display windows can be used: scroll windows (similar to a paper tape recorder, plotting curves with real-time data), number windows to display numerical data, button windows for on/off orders, etc. Using the mouse and a graphics-based interface with dialog windows, the user can select the pertinent windows, put them on any position of the screen, resize them, etc. until obtaining a complete application screen (Fig. 9 shows an example, with a combination of several types of windows). Then, the user assigns functions to each window. Clicking on a window, a form appears: the user fills this form to specify what the window will do, for instance to plot the real-time measurements of node 1 channel 3, together with the measurements of node 4 channel 9 for comparison purposes.

Once the application is defined, this definition can be saved as a description file (it can be used for re-design, including modifications, improvements, etc.). When a satisfactory definition is attained, the environment can be used to automatically generate a C program with the application. This program is clearly structured and documented, so any user with programming skills can easily add new features. The program uses the Quinn-Curtis graphical libraries [16]. With any popular C programming environment, such Borland C++, the application program can be compiled to get an executable code. This code is just what is needed to start the operation of the distributed system with the application.

The code generator works internally as follows. The description file is a set of data structures describing the windows on screen and the functional specifications given by the user. Each window has a specific behaviour (concerning both human interaction, and control and monitoring functions). The code generator analyses the description file and invokes pieces of source code (subroutines), building up a complete C program. This program is well-structured and has two parts: one is provided to handle the Quinn-Curtis libraries, and the other is devoted to configure and use the nodes.

7. Conclusion

Nowadays, monitoring and control applications are expanding to many industrial and daily life contexts. A good catalyst for this spread is to make easy the implementation of monitoring and control systems. This has been the objective of our research.

Our work starts recognising the interest of distributed systems, as an effective approach for cases where the complexity of the problem and/or the distances recommends the use of several interconnected nodes. Another aspect of interest is that embedded PCs offer the opportunity of having powerful not-expensive nodes.
This paper described a complete proposal for the easy implementation of distributed monitoring and control systems. The interconnection has been solved with robust standard techniques, based on modem and telephone means. The nodes are based on embedded PCs, and offer interesting autonomous capabilities. The main application creation is helped with a development software tool that can automatically generate the code.

This work can be easily extended to wireless communication with faster data rates, or to consider specific application fields where the power of embedded PCs can be exploited as an advantageous solution.

References


Jose Antonio de Frutos Redondo was born in Segovia, Spain. He received the Licentiate (1972) and the PhD (1978) degrees in Physics from the Universidad Complutense de Madrid, Spain; and the PhD (1998) from the Universidad de Alcalá, Spain. He is full professor with the Department of Computer Science of the Universidad de Alcalá since 1991. His research is related to applied automatic control and simulation and parallel computer architecture.

Jose Maria Giron-Sierra was born in Valladolid, Spain. He received the Licentiate (1972) and the PhD (1978) degrees in Physics from the Universidad Complutense de Madrid, Spain. He is full professor with the Department of Computer Architecture and Automatic Control from 1988. He has been the author of 60 publications in conference proceedings and journals. He holds two patents for a robot and a distributed control system. His research is related to applied automatic control and simulation: high-speed ships, airplanes and spacecrafts, cooperative robotics, logistics, process control, soft computing for optimisation. He is a member of IEEE, IFAC, and the Spanish Society of Pedagogy.