A System Architecture of Networked Pressure Sensors

MIROSLAV SVEDA*, RADIMIR VRBA**, PETR BENES***
*Dept. Computer Sci.&Engineering, **Dept. Microelectronics, ***Dept. Automation&Measurements
Brno University of Technology
Bozetechova 2
612 66 Brno
CZECH REPUBLIC
{sveda@dcse, vrbar@umel, benesp@dame}.fee.vutbr.cz
http://www.fee.vutbr.cz/{~sveda, ~vrbar, ~benesp}

Abstract: - This contribution deals with sensor networking concepts stemming from the IEEE 1451 smart transducer interface architecture. The paper reviews that approach from the viewpoint of coupling sensor-based appliances and the Internet. Kernel of the paper focuses on the application of the discussed architecture for a distributed pressure measurement system, stressing both intra-system and Internet connectivity issues. Next sections discuss in more detail the IEEE 1451 smart transducer interface for sensors and actuators as an emerging, standard-based networking framework.

Key-Words: - Decentralized systems, remote sensing, smart sensors, intranet, IEEE 1451, pressure analyzer

1 Introduction

Current decentralized systems of remote sensing can comprise various technologies, including such hot topics as Bluetooth [1] and Smart Dust [12]. More usual, industrial distributed systems typically encompass, at their lowest level, diverse digital sensor-to-controller connections based on low-level fieldbuses that constitute the bottom, vendor-dependent segments of hierarchical communications. That hierarchy includes higher-level fieldbus or LAN backbones and, more recently, an access to and from intranets/Internet. Consequently, such systems must comprise suitable interconnections of incident backbone and lower fieldbus segments, which intermediate not only top-down commands and bottom-up responses but also remote system management.

A special case of the previous problem appears connecting a fieldbus with sensors to an intranet. While for the majority of high-level fieldbuses the solutions are commercially available, they offer, as a rule, a proprietary approach. Similarly to fieldbus-to-fieldbus coupling, it is possible only to map the broad variety of concepts applicable to fieldbus-to-intranet interconnections.

Current industry is migrating away from proprietary hardware and software platforms mentioned above in favor of open and standardized approaches. Internet technologies such as Java, WWW, TCP/IP, and also Ethernet are rapidly becoming the platforms of choice for building next generation distributed measurement and control systems. The framework, which can support the current trend, is the IEEE 1451 smart transducer interface architecture that enables to unify not only interconnecting smart sensors with various fieldbuses but also direct coupling to the Ethernet-based intranets. The proposed standards include an object-oriented information model targeting software-based, network independent, transducer application environments with a unified digital interface and a communication protocol for accessing sensors and actuators.

2 Sensors-to-Intranet Coupling

To join fieldbus-based sensors to an Ethernet-compatible intranet, it is necessary to develop a full gateway that has to adhere to the adjacent fieldbus protocol profile on one side and to the selected Internet protocol profile on the other side, and has to translate between messaging protocols of application layers in both domains. While a concrete fieldbus

protocol suite usually predetermines the fieldbus side of the gateway architecture, the intranet side offers free choice among many possible profiles even if we focus our attention on WWW-based technologies, see e.g. [2], [9].

Evidently, the interconnections of sensors with Internet/intranets by mediating fieldbus systems result in multitude of proprietary solutions. Those interconnections cannot be standardized internationally because of the cancellation of the related IEC Fieldbus standardization initiative. Hence, that methodology appears useful only to meet special application or implementation requirements. On the other hand, there is a possibility how to manage that problem utilizing the sensor/actuator-based standardized approach mentioned above.

The IEEE 1451 consists of the family of standards for a networked smart transducer interface that include namely a smart transducer information model, 1451.1 [3], targeting software-based, network independent, transducer application environments, and a standard digital interface and communication protocol, 1451.2 [4], for accessing the transducer via the microprocessor modeled by the 1451.1. (The transducer information model called as network capable application processor, NCAP, is discussed in more detail in Section 4.) The next two standards, 1451.3 and 1451.4, extend the possible single-attached configurations to embedded distributed multidrop systems and to mixed-mode communication protocols for analog transducers.

The IEEE 1451 framework enables to develop unified interconnection to the Ethernet-based intranet. The following section reviews the use of the IEEE 1451 standards in a real application.

3 Pressure Analyzer

The pressure analyzer consists of a group of smart pressure sensors interconnected by the Ethernet-based intranet with supporting nodes, enabling also to join to components on various fieldbuses, see Fig. 1. The complete research task has been aimed at application-driven pressure measurement processes and tools starting with the study of various physical principles and designs of transducers up to the final data processing implementation including internal and external data communications.

3.1 Communication interface

One of the initial problems to be solved consisted in the selection of a unified communication interface. The first idea appeared to utilize the international standard IEC Fieldbus and to develop an IEC fieldbus-to-intranet gateway with a communication architecture fitting this application domain. When the IEC Fieldbus standardization initiative was cancelled, our research group focused attention on the IEEE 1451 standard. The final solution of that application respects IEEE 1451 in such a way that enables both direct coupling of sensors to intranet and to interconnected fieldbuses.

3.2 Smart sensor internal communication

The core components of the pressure measurement applications appear to be pressure sensors, in this case smart devices equipped with microprocessors. Each smart sensor contains one of two NCAP implementations based either on 16-bit ARM 7 or on 8-bit Rabbit microprocessors that both provide realization of the 1451.1 object model for intranet interconnectivity and the access to the 1451.2 interface for connecting to pressure sensors with reflected laser beam and diffractive lens.

Read-out sensing system applies a combination of two principles that provide both high precision and wide range pressure measurements. Large displacements of a membrane are measured by the position of the reflected focused laser beam. Small position changes are measured by one-side layer diffractive lens principle. Sensor output signal is conditioned in digital by ADuC812 one-chip microcomputer, which provides the IEEE1451.2 interface as one of its communication ports.

3.3 Intranet connectivity

The intranet side stems from a Java-based architecture with Web server in the role of a half gateway. The Java applet represents the client-side of the client-server communication model. The software half gateway provides the key server-side capabilities allowing Java clients to connect, subscribe, and communicate to the smart sensors. Java can directly support client-server application architectures as the core Java specification includes a TCP/IP networking API.

The developed Java applet uses the core java.net package to implement a client-server application distribution. This allows the Java applet client to access smart sensors and supporting nodes. The Java applet consists of a series of object classes, including multi-threaded applet environment, animation, and TCP/IP-based network client communications.

3.4 Fieldbus connectivity

The application architecture employs a configurable server implementing the functionality of a fieldbus gateway that enables to access various fieldbus-level components compatible to such protocols as CAN with DeviceNet or CANopen, Fieldbus Foundation, and LonWorks. Moreover, we consider the fieldbus-to-fieldbus interconnection techniques developed earlier [11], which can further extend the range of connectable devices.

This server has supported the reuse of previously developed sensors not only for launching the system implementation experiments and sensor testing purposes but also for real-life, concrete application development.

4 IEEE 1451 Network Architecture

To explain the principles of the sensor networking architecture described above, this section reviews the IEEE 1451.1 standard [3]. The 1451.1 information model [5] deals with an object-oriented definition of a network capable application processor, NCAP, which is the object-oriented embodiment of a smart networked device. This model includes the definition of all application-level access to network resources and transducer hardware.

4.1 Object model

The object model definition encompasses a set of objects classes, attributes, methods, and behaviors that provide a concise description of a transducer and a network environment to which it may connect. The standard brings a network and transducer hardware neutral environment in which a concrete implementation can be developed.

The standard uses block and base classes to describe the transducer device. The 1451.1 define four component classes offering patterns for one Physical Block, one or more Transducer Blocks, Function Blocks, and Network Blocks. Each block class may include specific base classes from the model. The base classes include Parameters, Actions, Events, and Files, and provide component class.

All classes in the model have an abstract or root class from which they are derived. This abstract class includes several attributes and methods that are common to all classes in the model and offer a definition facility to be used for instantiation and deletion of concrete classes. In addition, methods for getting and setting attributes within each class are also provided.

4.2 Block classes

Block classes form the major blocks of functionality that can be plugged into an abstract card-cage to create various types of devices. One Physical Block is mandatory as it defines the card-cage and abstracts the hardware and software resources that are used by the device. All other blocks and component base classes can be referenced from the Physical Block.

The Physical Block representing the card-cage contains all the logical hardware and software resources in the model. These resources determine the basic characteristics of the device being assembled. Information contained in the Physical Block as attributes include the manufacturer's identification, serial number, hardware and software revision information, and more importantly, data structures that provide a repository for other class components. As previously mentioned, the Physical Block is the logical container for all components in the device model; therefore, it must have access to and be able to locate all available resources instantiated by the device. The data structures provided by the Physical Block house pointers (Instance_ID) to these components in that way offer easy indirect access to them. To communicate to a device or a device object across the network when a remote NCAP requests an attribute from the Physical Block, that Physical Block has to resolve address queries from the network. For this purpose a hierarchical naming/addressing scheme is used based on unique Tags, i.e. ASCII descriptions of the block or component names, which can be concatenated together, to form fully qualified addresses. The Physical Block is the centralized logical connector or backplane that the other blocks plug into. Therefore,

the Physical Block must provide a Locate method to find other components in the system.

The Transducer Block abstracts all the capabilities of each transducer that is physically connected to the NCAP I/O system. During the device configuration phase, the description is read from the hardware device what kind of sensors and actuators are connected to the system. This information is used by the Physical Block to create and configure the related type of transducer block. The Transducer Block includes an I/O device driver style interface for communication with the hardware. The I/O interface includes methods for reading and writing to the transducer from the application-based Function Block using a standardized interface (i.e., io_read and io_write). The I/O device driver paradigm provides both plug-and-play capability and hot-swap feature for transducers. This means any application written to this interface should work interchangeably with multiple vendor transducers.

In a similar fashion the transducer vendors provide an I/O driver to the network vendors with their product that supports this interface. The driver is integrated with the transducer's application environment to enable access to their hardware. This approach is identical to the interface found in device drivers for UNIX.

The Function Block equips a transducer device with a skeletal area in which to place application-specific code. The interface does not specify any restrictions on how an application is developed. In addition to a State variable that all block classes maintain, the Function Block contains several lists of parameters that are typically used to access network-visible data or to make internal data available remotely. It means, any application-specific algorithms or data structures are contained within these blocks to allow separately for integration of application-specific functionality using a portable approach.

The Network Block is used to abstract all access to the network by the block and base classes employing a network-neutral, object-based programming interface. The network model provides an application interaction mechanism based on the remote procedure call (RPC) framework for client-server distributed computing settings. The RPC mechanism supports both a client-server and a publisher-subscriber paradigm for event and message generation. In support of these two types of application interaction, a communication model that stems from the notion of a port is defined in the specification. This means, if a block wishes to communicate with any other block in the device or across the network, it must first create a port that logically binds the block to the port name. Once enough information about addressing of the port is known, the port can be bound to a network-specific block address. At this point the logical port address has been bound to the actual destination address by the underlying control network technology. Any transducer application's use of the port name is now resolved to the endpoint associated with the logical destination. This scheme allows a late binding effect on application uses of the ports so that addresses are not hard-coded or dependent upon a specific architecture. The port capability is similar to the TCP/IP socket-programming interface where a socket is created and bound using an application specific port number and IP address. Once bound, the socket can be used for data transfer.

4.2 NCAP implementation

The above introduced object model has to be implemented so that the NCAP can provide all communication between each client and the connected sensor. Namely, the Network Block and supporting hardware must prop up the complete protocol stack of the adjacent network including a messaging protocol respected in frame of the application.

The pressure analyzer includes examples of two NCAP implementations by the ARM 7 and Rabbit microprocessors software for direct sensor-to-intranet coupling. The fieldbus gateway server demonstrates how to reuse various fieldbus-dependent sensors to be accessible from the intranet.

5 Related Work

The research group at the NIST [8] developed demonstration architectures for coupling 1451-compatible devices to fieldbuses or Internet-based intranets [5], [9], [10]. Those papers originally attracted our attention to the IEEE 1451 standards.

We are aware of announced commercial solutions both of the 1451.1 NCAP [6] and of the 1451.2-compatible pressure sensor [7]. Unfortunately, those

devices neither meet the pressure analyzer requirements specification nor are currently available.

6 Conclusion

This paper presents an application that can offer a design pattern for 1451 sensors-based embedded systems. It deals with IEEE 1451 smart transducer network interface mediating an access from the Internet to networked sensors. The kernel of the paper focuses on the pressure analyzer as a real-world application, stressing the communication architecture including intranet connectivity issues.

Acknowledgments

The authors appreciate contributions of their colleagues from the Department of Computer Science and Engineering, the Department of Microelectronics, the Department of Automatic Control and Measurement Techniques, and the Department of Telecommunications at the Brno University of Technology, namely Frantisek Zezulka, Vladislav Musil, Kamil Vrba, Daniel Becvar, Michal Strach and Michal Jurica to the presented work.

This research has been supported by Grand Agency of the Czech Republic under the contract GACR 102/00/0938 TLAKAN and by the Czech Ministry for Education in frame of the Research Intention JC MSM 262200012: Research in information and control systems.

References:

[1] Bluetooth Special Interrest Group, *The Official Bluetooth Website*, http://www.bluetooth.com/

[2] D. Bühler and W. Küchlin and G. Gruhler and G. Nusser, The Virtual Automation Lab -- Web Based Teaching Automation Concepts, *Proceedings 7th IEEE ECBS'2000 Conference*, Edinburgh, Scotland, IEEE Comp. Soc., 2000, pp.156-164.

[3] *IEEE P1451.1 D1.83, Draft Standard for a Smart Transducer Interface for Sensors and Actuators -- Network Capable Application Processor (NCAP) Information Model.* IEEE, New York, December 1996

[4] *IEEE P1451.2 D3.05, Draft Standard for a Smart Transducer Interface for Sensors and Actuators -- Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEEDS) Formats,* IEEE, New York, August 1997

[5] K.B. Lee and R.D. Schneeman, A Standardized Approach for Transducer Interfacing: Implementing IEEE-P1451 Smart Transducer Interface Draft Standards, *Proceedings Sensors Expo,* Philadelphia, Hermes Publishing, October 1996, pp.87-100.

[6] Microsmith Inc., *NetBoss, IEEE 1451.1 NCAP Network Control Module*, 2000, http://www.microsmith.com

[7] Moore Products Co., *IEEE 1451.2-Compatible Pressure Sensors*, 2000, http://www.mpco.com

[8] National Institute of Standards and Technology, *NIST IEEE 1451 Website, 2000,* http://www.motion.aptd.nist.gov

[9] R.D. Schneeman and K.B. Lee, *Multi-Network Access to IEEE P1451 Smart Sensor Information Using World Wide Web Technology,* NIST Interagency Report 6136, National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 1997.

[10] R.D. Schneeman, *Implementing a Standards-based Distributed Measurement and Control Application on the Internet,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 1999

[11] M. Sveda and R. Vrba and F. Zezulka, Coupling Architectures for Low-Level Fieldbuses, *Proceedings of the 7th IEEE ECBS'2000 Conference*, Edinburgh, Scotland, IEEE Comp. Soc., 2000, pp.148-155.

[12] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister, Smart Dust: Communicating with a Cubic-Millimeter Computer, *IEEE Computer,* Vol.34, No.1, 2001, pp.44-51.

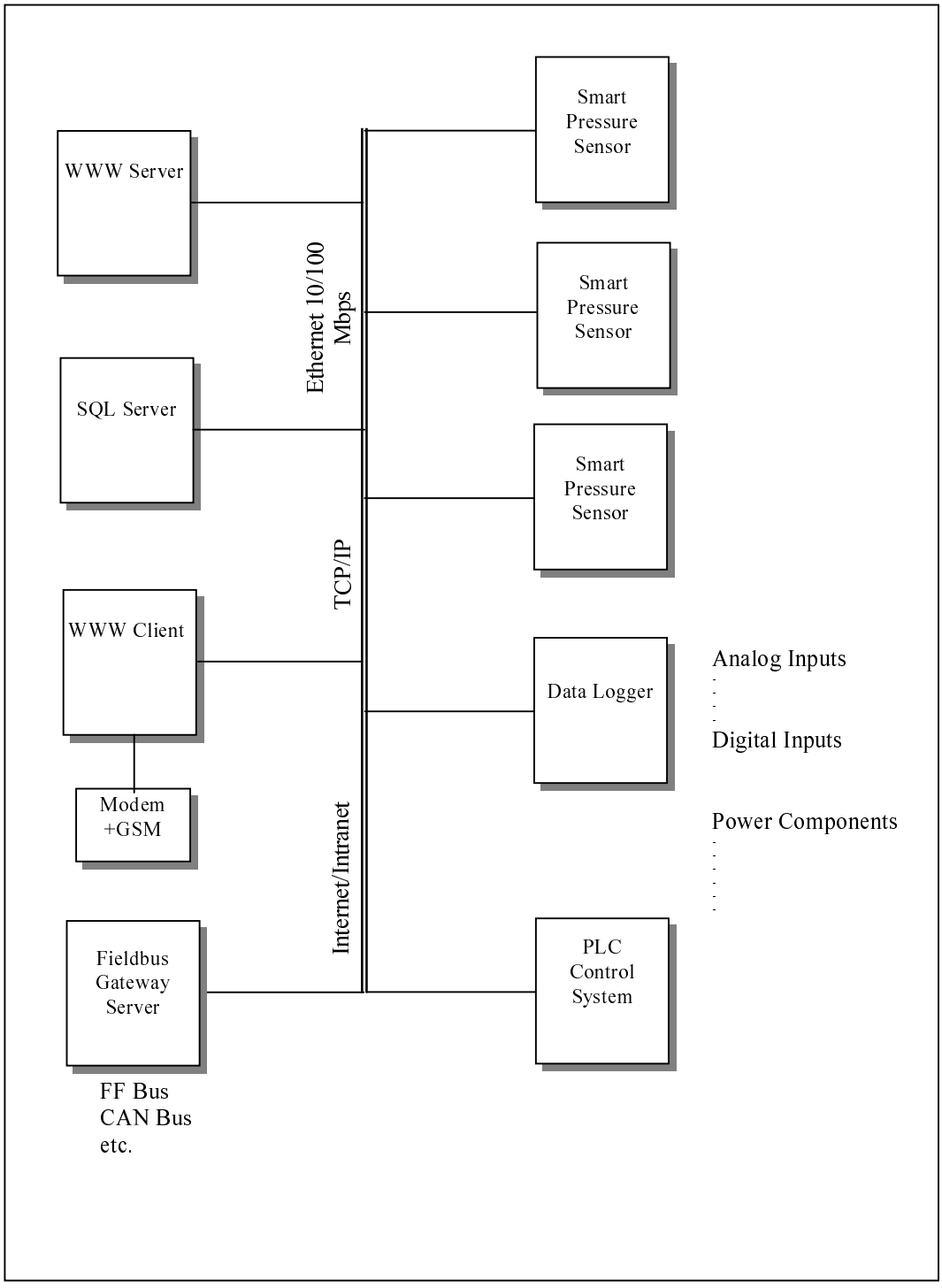

Fig. 1. Pressure analyzer