A Standardized Smart Transducer Interface

Wilfried Elmenreich and Roman Obermaisser
Institut für Technische Informatik
Technische Universität Wien
Karlsplatz 13, Vienna, Austria
{wil,romano}@vmars.tuwien.ac.at

May 8, 2002

Abstract

In December 2000 the Object Management Group (OMG) called for a proposal of a smart transducer interface. In response a new standard has been proposed that comprises a time-triggered transport service within the distributed smart transducer subsystem and a well-defined interface to a CORBA environment.

This smart transducer standard has been adopted by the OMG in January 2002. The smart transducer interface and the time-triggered transport service have been implemented in the time-triggered fieldbus protocol TTP/A.

This paper examines the strengths and weaknesses of the smart transducer interface and presents two case studies. The first case study describes a demonstrator with a robot arm that is instrumented by smart transducers and a TTP/A fieldbus. The second case study is an autonomous mobile robot instrumented by a TTP/A cluster that orientates itself via a set of heterogenous sensors and decides, based on this sensor data, which way to go.

1 Introduction

The design of the network interface for a smart transducer is of great importance. Transducers can come in a great variety with different capabilities from different vendors. A smart transducer interface must thus be very generic to support all present and future types of transducers. However, it must provide some standard functionalities to transmit data in a temporal deterministic manner in a standard data format, provide means for fault tolerance, and enable a smooth integration into a transducer network and its application.

A smart transducer interface should conform to a world-wide standard. Such a standard for a real-time communication network has been long sought, but efforts to find one agreed standard have been hampered by vendors, which were reluctant to support such a single common standard in fear of losing some of their competitive advantages [1]. Hence, several different fieldbus solutions have been developed and promoted. Some of these existing solutions have been combined and standardized. In 1994, the two large fieldbus groups ISP (Interoperable Systems Project supported by Fisher-Rosemount, Siemens, Yokogawa, and others) and the WorldFIP (supported by Honeywell, Bailey, and others) joined to form the Fieldbus Foundation (FF). It is the stated objective of the FF to develop a single interoperable fieldbus standard in cooperation with the International Electrotechnical Commission (IEC) and the Instrumentation Society of America (ISA).

The IEC worked out the IEC 61158 standard. It is based on eight existing fieldbus solutions. However, the IEC fieldbus draft standard was not ratified at the final approval vote, following a set of controversies [2]. The IEC 61158 has the great disadvantage that it still keeps a diversity of eight different solutions.

The ISA, which developed the SP50 standard and

IEC committees met jointly to make the development of an international standard possible. ISA SP50 was the same committee that introduced the 4-20 mA standard back in the 1970s.

Meanwhile, other standards for smart transducers were developed. The IEEE 1451.2 [3] standard deals with the specification of interfaces for smart transducers. An idea proposed by this standard is the specification of electronic data sheets to describe the hardware interface and communication protocols of the smart transducer interface model [4].

In December 2000 the Object Management Group (OMG) called for a proposal of a smart transducer interface (STI). In response a new standard has been proposed that comprises a time-triggered transport service within the distributed smart transducer subsystem and a well-defined interface to a CORBA (Common Object Request Broker Architecture) environment. The key feature of the STI is the concept of an Interface File System (IFS) that contains all relevant transducer data. This IFS allows different views of a system, namely a real-time service view, a diagnostic and management view, and a configuration and planning view. The interface concept encompasses a communication model for transparent time-triggered communication. This STI standard has been adopted by the OMG in January 2002.

It is the objective of this paper to examine the strengths and weaknesses of the STI standard and to present two case studies showing the capabilities of the smart transducer interface.

The rest of the paper is organized as follows:

The following section explains the conceptual model and architecture of the STI. Section 3 introduces the properties of the IFS. Section 4 depicts the proposed time-triggered fieldbus protocol. In section 5 we will describe two case studies, that implemented smart transducer networks based on the STI. Section 6 investigates on the strengths and weaknesses of the STI standard. The paper is concluded in section 7.

2 Conceptual Model

This section will introduce the conceptual model of the OMG smart transducers interface [5].

On a abstract level, the purpose of a real-time smart transducer interface is the timely exchange of "observations" of real-time entities between the engaged subsystems across the provided interfaces. A real-time entity is a state variable of interest that has a name and a value at a particular instant. An observation [6] is thus an atomic triple:

$$<\text{name, observation instant, value}>,$$

where name is an element of the common name space of real-time entities, the observation instant is a point in the "time space" and value is an element of the chosen value domain. An observation expresses that the referenced real-time entity possessed the stated value at the indicated instant.

A smart transducer (ST) system consists of several clusters with transducer nodes connected to a bus. Each cluster is connected to a CORBA gateway via a master node. One CORBA gateway can interface up to 250 clusters. The master of each cluster is connected to the CORBA gateway through a real-time communication network, which provides a synchronized time to each master. Each cluster can contain up to 250 STs that communicate via a cluster-wide broadcast communication channel. There may be redundant shadow masters to support fault tolerance. One active master controls the communication within a cluster (in the following sections the term master refers to the active master unless stated otherwise). Since smart transducers are controlled by the master, they are called slave nodes. Figure 1 depicts an example for such a smart transducer system.

Access to the smart transducer data is achieved by assigning three different interfaces to each ST node:

DM interface: This is a *diagnostic and management* interface. It establishes a connection to a particular ST node and allows reading or modifying of specific IFS records. Most sensors need parameterization and calibration at

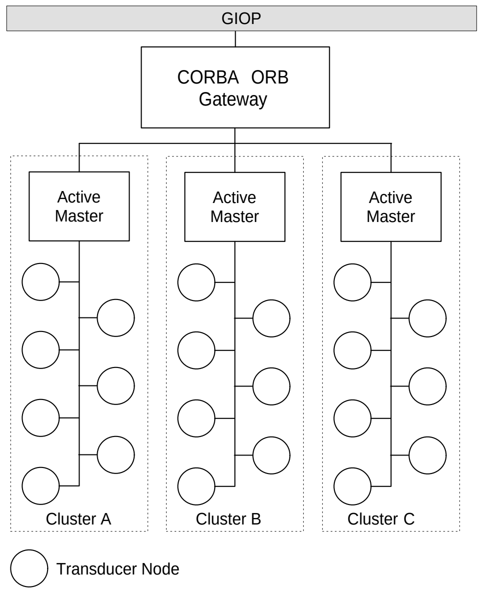

Fig. 1: Multi-Cluster Architecture with CORBA Gateway

startup and continuously collect diagnostic information to support the maintenance activities. For example a remote maintenance console can request diagnostic information from a certain sensor.

CP interface: The *configuration and planning* allows the integration and setup of newly connected nodes. It is used to generate the "glue" in the network that enables the components of the network to interact in the indented function. Usually, the CP interface is not time-critical.

RS interface: The *real-time service* interface performs a periodic communication with predictable timing behavior among the ST nodes. Communicated data is usually data from sensors and for actuators. This view employs sensors for producing periodic observations of real-time entities in the environment. This view supports the normal operation of the sensor. For example, a temperature sensor periodically sends the observed and locally preprocessed sensor value to the temporal firewall of the master. Since in TTP/A the time interval between sensing the environment and presenting the sensor value at the temporal firewall [7] of the master is known a priori it is possible to perform a feed forward state estimation of the sensor value at the sensor node in such a way, that the delivered sensor value is a good estimate of the real-time entity's actual state at the point in time of delivery.

Although all transducer nodes are built as smart transducers and contain a physical sensor or actuator, a microcontroller, and a network interface, the hardware requirements for the ST interface are very flexible. The STI supports low-cost implementations of smart transducers, by allowing optional implementation of standard features. Thus, it is possible to fit a minimum STI implementation on an embedded microcontroller with 2k flash memory and 64 bytes of RAM memory [8].

3 Interface File System

The information transfer between a smart transducer and its client is achieved by sharing information that is contained in an internal interface file system (IFS), which is encapsulated in each smart transducer.

A time-triggered sensor bus will perform a periodical time-triggered communication to copy data from the IFS to the fieldbus and write received data into the IFS. Thus, the IFS is the source and sink for all communication activities. Furthermore, the IFS acts as a temporal firewall that decouples the local transducer application from the communication activities.

It is the task of the communication protocol to keep consistency among the local copies of the IFS data elements. A predefined communication schedule defines time, origin, and destination of each protocol communication. The instants of updates are specified a priori and known by the communicants. Thus, the IFS acts as a *temporally specified interface* that decouples the local transducer application from the communication task.

Each transducer can contain up to 64 files in its IFS. An IFS file is an index sequential array of up to 256 records. A record has a fixed length of four bytes (32 bits). An IFS record is the smallest addressable unit within a smart transducer system. Every record of an IFS file has a unique hierarchical

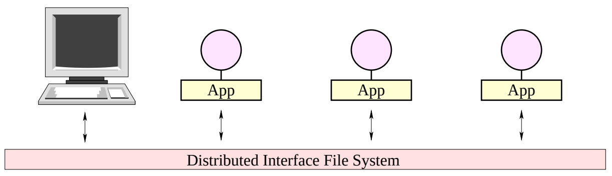

Fig. 2: Logical Network Structure

address (which also serves as the global name of the record) consisting of the concatenation of the cluster name, the logical name, the file name and the record name.

The IFS provides a unique address scheme for transducer data, configuration data, self-describing information and internal state reports of a smart transducer [9].

Besides access via the master node, the local applications in the smart transducer nodes are also able to execute a clusterwide application by communicating directly with each other. Figure 2 depicts the network view for such a clusterwide application.

As depicted in Figure 3, the IFS of each smart transducer node can be accessed via the RS interface, the DM interface and the CP interface for different purposes. All three interfaces are mapped onto the fieldbus communication protocol, but with different semantics regarding timing and data protection.

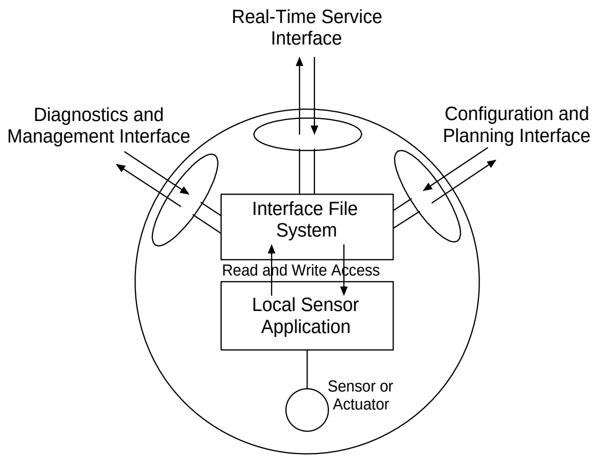
Real-Time Service Interface
Diagnostics and Management Interface
Configuration and Planning Interface
Interface File System
Read and Write Access
Local Sensor Application
Sensor or Actuator

Fig. 3: Three Interfaces to a Smart Transducer Node

4 Fieldbus Communication Protocol

A time-triggered transport service following the specification of the STI has been implemented in the time-triggered fieldbus protocol TTP/A.

The bus allocation is done by a Time Division Multiple Access (TDMA) scheme. Communication is organized into rounds consisting of several TDMA slots. A slot is the unit for transmission of one byte of data. Data bytes are transmitted in a standard UART format. Each communication round is started by the master with a so-called fireworks byte. The fireworks byte defines the type of the round.

The protocol supports eight different firework bytes encoded in a message of one byte using a redundant bit code [10] supporting error detection.

Generally, there are two types of rounds:

Multipartner round: This round consists of a configuration dependent number of slots and an assigned sender node for each slot. The configuration of a round is defined in a datastructure called “RODL” (ROund Descriptor List). The RODL defines which node transmits in a certain slot, the operation in each individual slot, and the receiving nodes of a slot. RODLs must be configured in the slave nodes prior to the execution of the corresponding multipartner round. An example for a multipartner round is depicted in Figure 4.

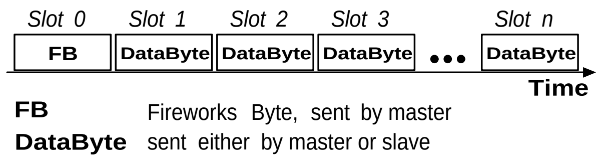
Slot 0
Slot 1
Slot 2
Slot 3
Slot n
FB
DataByte
DataByte
DataByte
DataByte
Time
FB Fireworks Byte, sent by master
DataByte sent either by master or slave

Fig. 4: A TTP/A Multipartner Round

Master/slave round: A master/slave round is a special round with a fixed layout that establishes a connection between the master and a particular slave for accessing data of the node’s IFS, e. g. the RODL information. In a master/slave round the master addresses a data record in the hierarchical IFS address and specifies an action like reading, writing or executing on that record.

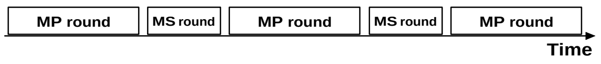

Fig. 5: Recommended TTP/A Schedule

The master/slave rounds establish the DM and the CP interface to the transducer nodes. The RS interface is provided by periodical multipartner rounds. Master/slave rounds are scheduled periodically between multipartner rounds as depicted in Figure 5 in order to enable maintenance and monitoring activities during system operation without a probe effect.

5 Case Study Implementations

5.1 Robot Arm

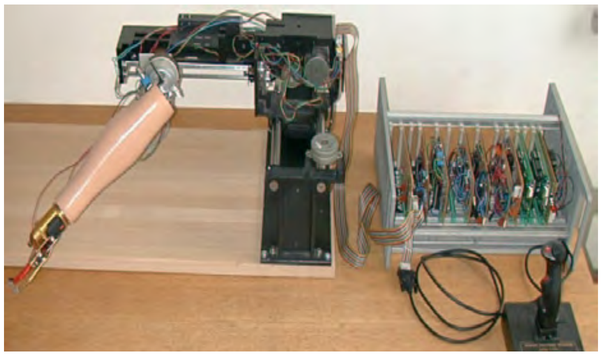

Fig. 6: Robot Arm

As a demonstrator for the STI we built a system with a robot arm [11]. At the application level a human operator can control a prosthetic arm mounted on top of a linear thrust unit (See Figure 6). Simplicity of control for the user is established by the presence of intelligence in the demonstrator. Smart sensors yield information about the environmental conditions allowing avoidance of operating errors and obtaining precise control. Pressure sensors are present for determining the required grip force to grasp an object. No intervention of the human operator is needed to avoid slipping of an object. Motor actuator nodes implement trapezoid excitation for handling of inertia. The demonstrator is equipped with an angle sensor to allow limiting the opening of the elbow.

The demonstrator was also intended to investigate partitioning of nodes into distinct clusters. The resulting intercluster communication was required to preserve temporal predictability and capabilities for monitoring, maintenance and configuration.

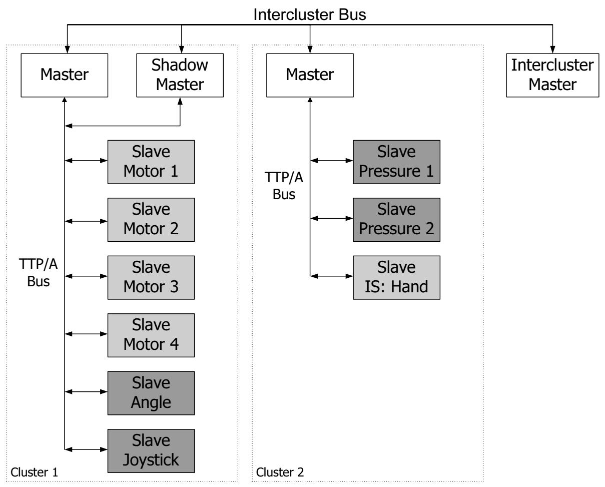

Fig. 7: Architecture of Robot Arm Transducer Network

The demonstrator consists of two clusters (See Figure 7). The first cluster contains the nodes for controlling the motors of the linear thrust units, the elbow and the wrist. The nodes for retrieving the current angle of the elbow and the joystick commands are also placed in this cluster. A shadow master can take over control in case the primary master fails. Both masters are connected to the intercluster bus and act as intermediate systems. In addition to their TTP/A master role, they are slaves of a time-triggered backbone bus. The second cluster contains a node behaving as an interface system for integrating the prosthetic hand into the demonstrator. Two nodes equipped with pressure sensors obtain measurements for grasping objects intelligently.

5.2 Autonomous Mobile Robot

Another implementation of the STI is shown by a model car, that acts as an autonomous robot with sensory inputs [12]. Seventeen nodes were used for building this mobile robot ("smart car"). Some of these nodes are implemented on very small MCUs to demonstrate the possibility of cheap slaves. Other nodes should demonstrate the possibility of complex features like plug & play or reconfiguration.

The model comprises a smart car equipped with a suit of pivoted distance sensors, an electric drive and a steering unit. Distance sensors, servo motors for sensor pivoting, driving and steering units are all separate smart transducer nodes. Each node is implemented on a low-cost microcontroller and equipped with an STI.

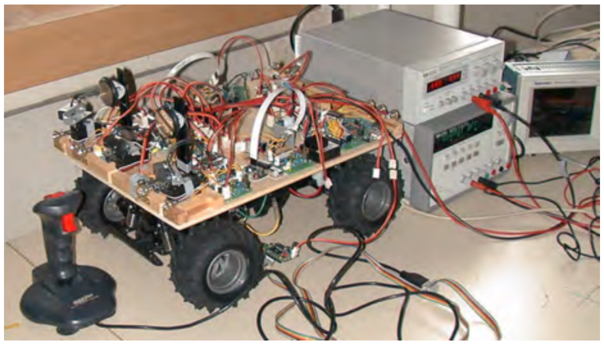

Fig. 8: Smart Car

The STI supports the integration of smart transducer nodes with a predictable timing behavior [13]. It is possible to add extra “light” nodes to the car during operation.

Figure 8 depicts the functionality of the smart car. In order to achieve short efficient operation two different operation modes are defined. The STI standard supports up to 6 user-definable independent communication modes, which support applications running different modes.

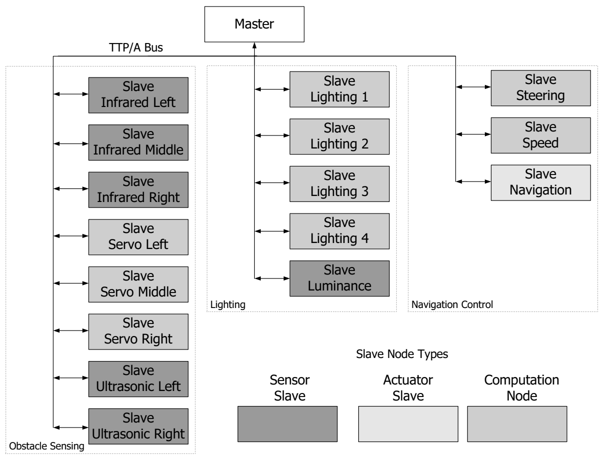

Fig. 9: Architecture of Smart Car Transducer Network

As long as no obstacles are detected within the sensors’ range the car operates in “rabbit mode”. In this mode the car drives straight forward at full speed and two infrared sensors are aimed slightly outward the driving direction. The main detection of obstacles relies on two ultrasonic sensors. These are capable to report obstacles straight ahead of the car within a range of about 5m.

In case an obstacle is detected the car switches to “turtle mode”. In this mode the car uses a communication schedule where all infrared sensors and pivot servos are serviced. The distance sensors are swivelled around by servo motors so that they are able to scan the area in front of the robot. The sensors generate a value that corresponds to the distance of the object they are aimed at. The data stream provided by the distance sensors is taken over by a data processing node that fuses the perceptions from the distance sensors and the directions they are aimed at with a model of the robot environment. In this model the shapes of obstacles are stored and assigned with a probability value, that decreases with the progression of time and increases when the object is re-scanned. From this data a navigation node calculates the speed and the direction to provide this information to the speed and steering nodes.

6 Discussion

One requirement stated in the call for proposal by the OMG was real-time capability of the smart transducer interface.

The STI supports hard real-time communication by introducing a time-triggered communication scheme, that is a priori specified before the RS interface of the system is used.

Generally, time-triggered systems require an increased effort in the design phase of the system, but provide an easier verification of the temporal correctness [14]. Since time-triggered systems are designed according to the principle of resource adequacy [15], it is guaranteed that sufficient computing resources are available to handle the specified peak load scenario. On the other hand, time-triggered systems are often blamed for their bad flexibility. The STI overcomes this limitation by introducing means to configure the interaction of the components via the CP interface.

The RS interface provides composability, guaranteed timeliness, and hides components' internals. The DM and CP interfaces involve inherently event-triggered activities, which require an event-triggered communication service. These interfaces cannot invalidate the temporal behavior of the RS interface and support full access to component internals – as required by a maintenance engineer.

The specification of interfaces should be complete and of minimal cognitive complexity. Cognitive complexity can be minimized by restricting interactions via interfaces and by providing access restrictions. The kind of information that must be available via an interface depends on the purpose of the particular interface. For example, a properly designed operational interface hides component internals, thereby allowing a component to form a meaningful abstraction. The corresponding operational interface specification stipulated during architecture design should incorporate a precise specification of a component's inputs and outputs in both the temporal and value domain. A maintenance engineer on the other hand, might require access to intermediate computational results for locating the origin of an incorrect system behavior.

The smart transducer interface (STI) standard meets the requirement for complete interfaces of minimal cognitive complexity by introducing three different types of interfaces of a component. The separation into RS, DM and CP interface is done according to the interface purpose, the necessary level of visibility of component internal information, and the type of the temporal interaction patterns. Such a separation minimizes complexity in contrast to a universal interface incorporating support for all possible interactions.

The STI standard also specifies the provision of the three interfaces through a CORBA server. However, currently there is no CORBA architecture for effectively supporting the temporal requirements to establish the RS interface. Current priority-based approaches like Real-time CORBA [16] require complete knowledge about all other service requests and their corresponding priority values in the whole CORBA network. However, the availability of a global notion of time allows to record the instant of the acquisition of a real-time entity's state in each observation.

7 Conclusion

We implemented two case studies of the STI. The first case study comprises a demonstrator with a robot arm that is instrumented by a smart transducer network partitioned into two clusters. The second case study is an autonomous mobile robot, that shows the integration of new nodes and efficient communication despite of static communication schedules.

The results show, that the STI standard is an interesting option for various sensor network applications. The STI provides many features that are required by a fieldbus applications for automotive or automation industries.

Supported features are the real-time capability, the encapsulation of the node's internals, and a universal address space with the interface file system. The STI can be implemented on low-cost Commercial-off-the-Shelf (COTS) hardware and supports various bus media types.

8 Acknowledgments

This work was supported in part by the Austrian Ministry of Science, project TTSB and by the European IST project DSoS under contract No IST-1999-11585.

References

[1] J. J. Pinto. A neutral instrumentation vendor's perspective. *ISA Proceedings '94 and Intech July '95*, July 1995.

[2] P. Noury. WorldFIP, IEC 61158 and the internet: A new look at fieldbuses, 1999. Available at http://www.worldfip.org/noury02.html.

[3] P. Conway, D. Heffernan, B. O'Mara, P. Burton, and T. Miao. IEEE 1451.2: An interpretation and example interpretation. *Proceedings of the Instrumentation and Measurement Technology Conference*, pages 535–540, 2000.

[4] L. H. Eccles. A brief description of IEEE P1451.2. *Sensors Expo*, May 1998.

[5] Objective Interface Systems, TTTech Computertechnik, and VERTEL Corporation. Smart transducers interface. *OMG TC Document orbos/2001-06-03*, July 2001. Supported by Technische Universität Wien. Available at http://www.omg.org.

[6] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[7] H. Kopetz and R. Nossal. Temporal firewalls in large distributed real-time systems. *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FT-DCS '97)*, pages 310–315, 1997.

[8] C. Trödhandl. Architectural requirements for TTP/A nodes. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2002.

[9] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2), March 2001.

[10] W. Haidinger and R. Huber. Generation and analysis of the codes for TTP/A fireworks bytes. Research Report 5/2000, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2000.

[11] R. Obermaisser. Design and implementation of a distributed smart transducer network. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2001.

[12] W. Elmenreich, W. Haidinger, H. Kopetz, T. Losert, R. Obermaisser, M. Paulitsch, and C. Trödhandl. A smart sensor LIF case study: Autonomous mobile robot. *DSoS Project (IST-1999-11585) Deliverable PCE3*, Apr. 2002.

[13] Wilfried Elmenreich, Wolfgang Haidinger, Philipp Peti, and Lukas Schneider. New node integration for master-slave fieldbus networks. In *Proceedings of the 20th IASTED International Conference on Applied Informatics (AI 2002)*, pages 173–178, Feb. 2002.

[14] Hermann Kopetz. Should responsive systems be event-triggered or time-triggered? *Institute of Electronics, Information, and Communications Engineers (IEICE) Transactions on Information and Systems*, E76-D(11):1325–1332, 1993.

[15] H. W. Lawson. *Parallel Processing in Industrial Real-Time Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[16] D. C. Schmidt and F. Kuhns. An overview of the real-time CORBA specification. *IEEE Computer*, 33(6):56–63, 2000.