IMPLEMENTATION OF AN EVENT-TRIGGERED SMART SENSOR NETWORK ARCHITECTURE BASED ON THE IEEE 802.15.4 STANDARD

Erico Meneses Leão [*] **Luiz Affonso Guedes** [*]
Francisco Vasques [**]

[*] *Department of Computer Engineering and Automation - UFRN - Natal - Brazil*
[**] *Department of Mechanical Engineering - University of Porto - Porto - Portugal*

Abstract: A smart transducer is the integration of a sensor/actuator element, a processing unit, and a network interface. Smart sensor networks are composed of smart transducer nodes interconnected through a communication network. This paper proposed an event driven a smart sensor network architecture (asynchronous data) and its respective implementation based in the IEEE 802.15.4 standard. The events are derived from a data compression algorithm embedded into the smart sensor, which compresses data from the sensor. The architecture also supports configuration and monitoring activities of all distributed system.

Keywords: smart sensor, network, architecture, events, IEEE 802.15.4, compression algorithm.

1. INTRODUCTION

Automation activities are essential for the competitiveness increase in all industrial sectors. From a systemic approach, industrial automation can be characterized as a set of techniques that enable the construction of active subsystems with the capability to interact with the industrial processes for control, monitoring, and supervision proposes.

In the last few decades, the automation technologies have been evolving from a strongly centralized technology to an essentially distributed technology. Within this new approach, components are interconnected by digital communication networks. Thus, the traditional sensors based in the 4-20mA standard are being replaced by digital devices. These devices may have sensors, actuators, and control functionalities and are endowed of digital processors and communication systems. Within this context, a smart sensor is defined as the integration of an analog or digital sensor or an actuator element, a processing unit, and a network interface (Elmenreich, 2006).

In this way, the architectures of the lowest level of industrial automation are characterized for using a set of the smart transducers, usually connected through a communication network with real time properties. The distributed approach provides a significant improvement in the flexibility and scalability aspects of the industrial processes; however, it also brought new scientific and technological challenges, such as the need for new models and algorithms for real time and safe communications, considering financial and environmental restrictions.

The smart sensor design must deal with interchanges among devices, and also with interoperability and availability of information in real time. The networked operation of a smart sensor that uses standardized interfaces allows sharing information and resources. Thus, it allows the integration of all control and supervision processes, for example.

The objective of this paper is to propose an event-triggered smart sensor network architecture and its implementation based IEEE 802.15.4 standard (IEEE 802.15.4 Standard, 2006). This Architecture is similar to the OMG (*Object Management Group*) standard (Kopetz and Wien, 2003). However, differently to the original time-triggered OMG proposal, the proposed approach uses an asynchronous event-triggered mechanism to transmit the data from the smart sensors. This asynchronous behavior is a consequence of the implementation of a data compression algorithm into the smart sensors, which sends only the relevant data from the raw data mass. Thus, the proposed event-triggered approach saves an important amount of communication bandwidth.

The rest of the paper is organized as follows: Section 2 describes the proposed event-triggered smart sensor network architecture and its characteristics. In order to validate the proposed architecture, the Section 3 presents a prototype implementation using the IEEE 802.15.4 standard and some experimental results that were obtained from the prototype. The paper is concluded in Section 4.

2. EVENT-TRIGGERED SMART SENSOR NETWORK ARCHITECTURE

The proposed event-triggered smart sensor architecture was briefly introduced in (Leão *et al.*, 2007). In this paper, a more detailed presentation is made. It is based on the OMG's (*Object Management Group*) standard. The choice for this standard was motivated by its simple and well defined data access interface. However, differently to the time-triggered OMG standard (Kopetz and Wien, 2003), the proposed architecture follows an event-triggered approach. That is, the smart sensors send through the network only the relevant information about the process. The transfer of this information is triggered by asynchronous events.

Figure 1 shows the basic components of the proposed architecture. This architecture is composed by *clusters*. In turn, each *cluster* can have one master node and up to 255 slave nodes, which are interconnected by a field network. The master node is a most powerful processing device and it is designed to manage the cluster. It can also communicate with other master nodes through a supervision network. There may be redundant shadow masters to support fault tolerance. Thus, it is the responsible for management, control, and configuration activities throughout the system.

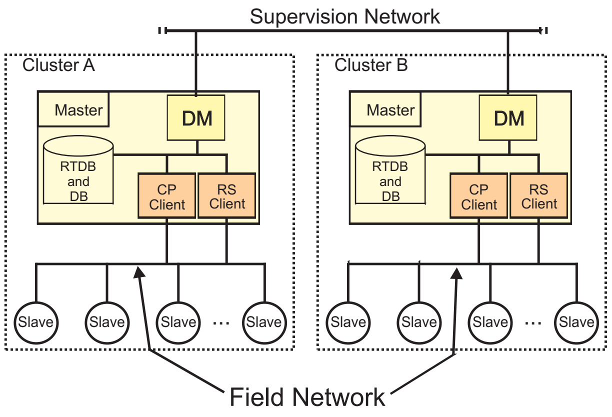

Fig. 1. Event-Triggered Smart Sensor Architecture.

2.1 Proposed Master Node Structure

The master node has two database types: a real time database (RTDB) and a traditional database (DB). The RTDB has to guarantee the temporal deadlines of its transactions. Thus, some data will be valid just for a specific time interval (Ramamritham, 1993). The master node logs in its databases the most relevant information from the smart sensors. These databases are accessed through the diagnostic and maintenance interface (DM) using the supervision network. The master node accesses information from the slave nodes using the RT client (real time client) and the CP client (configuration and planning client).

2.2 Proposed Slave Node Structure

A slave node requires two types of interfaces for accessing its data, a compression algorithm and a *buffer* to store temporary data. Such temporary data will be applied to the data compression algorithm, as shown in Figure 2. The compression algorithm is responsible for selecting the relevant data information, and thus it generates an asynchronous flow of data. This behavior leads to unpredictable timing intervals between consecutive data transfers from the sensor. However, the time of sending a datum is not totally unpredictable, due to the minimum and maximum time provided the compression algorithm to the transmission, as will be shown ahead. So, an event-triggered approach would be efficient for the communication of these control-related data.

The smart sensor will have a larger autonomy, as it will send only the relevant data. Therefore, there is a significant reduction of the transferred data,

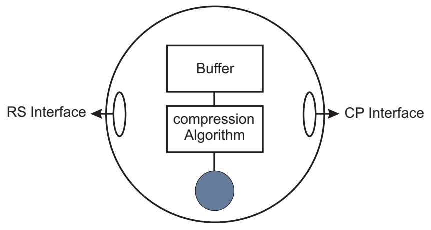

Fig. 2. Smart Transducer with a Embedded Compression Algorithm.

resulting in a smaller bandwidth utilization. This way, it is possible to connect a larger amount of smart sensors to the network. However, the smart sensors become more complex and therefore demand a larger processing capability. Nevertheless, with the growing technological progress it is possible to design low-cost smart sensors with high processing capabilities. Being so, the embedded compression algorithm is one of the research targets that must be addressed for the architectural network of event-triggered smart sensors.

Sensors require two types of interfaces to access the data (RS and CP), where the communication is supported by two different communication models (*publisher-subscriber* and *client-server*):

- **RS interface -** real-time service interface. It is used to transfer real-time data to the cluster. The data generated by the smart sensors will be published in the net and consumed by the functions of the system.
- **CP interface -** configuration and planning interface. Through this interface it is possible to identify new nodes connected to the network, to transfer new configuration parameters to the sensor, as values of compression deviation, maximum and minimum time for the compression algorithm, besides information as the identification of the sensor in the distributed system.

2.2.1. Publisher-Subscriber Model The communication model used by the RS interface is the *real-time publisher-subscriber model* (RTPS). This model of exchanging data favors the message exchange with time parameters amid devices between two entities: the publisher, responsible by sending the messages, and the subscribers, responsible for consuming these messages, in case they interest them (Dolejs *et al.*, 2004; Ocera, 2002). The messages sent by a slave node through the RS interface will be consumed by the various functions of the system concerned by such data. These messages can be send as commands to the actuators or stored in databases in accordance with their requisites; they can be real-time databases (RTDB) or traditional databases (DB), as shown in Figure 3.

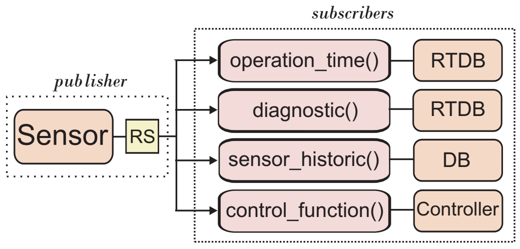

Fig. 3. *Publisher-Subscriber* Model for Proposed Architecture.

The data packets sent by smart sensors through the RS interface contain various fields with important information from the smart sensors, which will be consumed by the system functions to different finalities. For example, the *operation_time()* function will access the TIME field and will store this datum in the real-time database. Such function informs the supervision network about the functioning time of a determined smart sensor. The *sensor_historic()* function will access the information in the DATA field, in order to make available the historic file of flags for a smart sensor. The packet's TAG field is responsible for identifying the sensor; the TYPE field by the identification of the type of the packet to be transferred, while the QUALITY field will contain information about the quality flag of the datum (good, regular, bad, not determined). The CRC field is responsible for controlling the packet's error.

2.2.2. Client-Server Model The CP interface is accessed directly by the master node, which can perform configuration activities, can exchange parameters and can reconfigure of new nodes (Figure 4). As an example of the configuration of a new node, a master node through its function *new_device()* stays monitoring the system in order to search new nodes connected to the network. Thus, a new smart sensor will ask its inclusion in the system and will be given by the master node, through the function *configure()*, the necessary parameters to start functioning in the network. From this point onwards, after confirming the received parameter (*confirm()*) and having received the authorization to transfer its data (*start()*), the sensor will start sending its data packets through the RS interface.

Figure 5 shows the reconfiguration procedure of a smart sensor. The master node using the *request_reconfigure(ID)* function reconfigures the parameter of a particular slave node. Then, the slave node is authorized to send its packages.

Some important system functions are presented as follows:

- `request_inclusion()`: slave node requisites to master node its inclusion in the network.

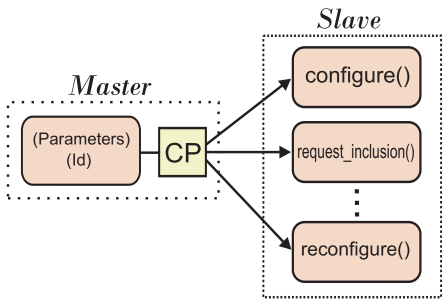

Fig. 4. *Client-Server* Model for Proposed Architecture.

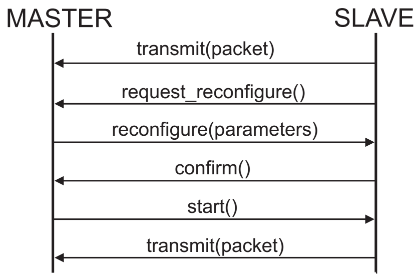

Fig. 5. Reconfiguration Proceeding of a Slave Sensor.

- `configure()`: master node sends configuration parameters of a new slave node.
- `reconfigure()`: master node sends reconfiguration to a slave node.
- `transmit()`: slave node sends a determined packet to the network.
- `operation_time()`: responsible for the time functioning o f a smart sensor.

3. CASE STUDY

This section presents a case study implementation of the proposed event-triggered smart sensor network architecture. The implementation uses the IEEE 802.15.4 (IEEE 802.15.4 Standard, 2006) standard as communication infrastructure. The prototype has been implemented upon the *Freescale Semiconduction* development kit.

3.1 Implementation based on IEEE 802.15.4

The IEEE 802.15.4 standard defines the physical and MAC layers for low cost, low power, and low rate devices (Baronti *et al.*, 2007). The embedded software was developed in accordance to *SMAC* (*Simple MAC*) premisses.

3.2 Definition of the Test Environment

Figure 6 shows the environment defined to experimentally assess this architecture, where it is defined two slave smart nodes and one master smart node connected through a field network. The supervision system can manage the smart sensor network through the master node.

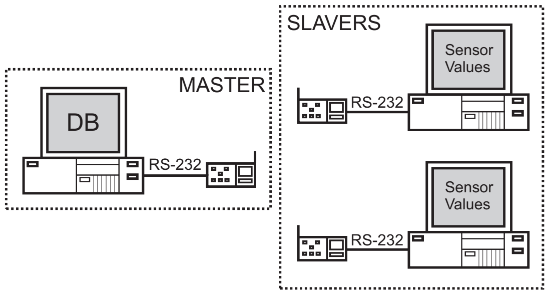

Fig. 6. Structure of a Smart Sensor.

For our application, a real sensor has been emulated using a PC equipped with a serial port and the network interface has been a wireless communication board of *Freescale.*

To implement the emulation process of the real sensors, we used two databases content records extracted from a real-time gas distribution monitoring system; specifically, we choose an outlet variable, with 10000 records. For testing effects, a program written in *ANSI C* was developed, it is capable to read each record from the database and to send it through RS-232 port (serial input/output) for the wireless communication board of *Freescale.*

The master node, on the other hand, is composed of a *Freescale* wireless communication board connected in a PC equipped with a software able to store the received values by master board in the database and generates the curve of the relevant values for each slave sensor connected upon the field network.

3.3 Definition of the Prototype Functions

The slave nodes support the *plug-and-play* paradigm. This paradigm is easily implemented because each slave node has an only MAC address as its identification on the network. Thus, the master node can detect the slave nodes in automatic way. When a slave node is connected to the network it requires to the master node its configuration. Then the master node reply with the following parameters: sensor ID, minimum and maximum time and compression deviate for the compression algorithm.

All the slave nodes have an implementation of the compression algorithm *Swinging Door* (Bristol, 1990). Thus, when a new data is generated, the compression algorithm decides if it is relevant or not, then this data can be send or not to the master board. To send a data to master node, the slave node must build a packet with the following fields: sensor identification (ID), current time and value. Then, when the master node receives any data from the slave node, it must store that data in its database for the activities of supervision network.

Another important function implemented in the prototype is the reconfiguration of a slave sensor. For example, through the master node, it is possible to reconfigure the compression algorithm of any smart sensor connected to field network.

3.4 Test Scenarios

To validate the prototype, we defined two test scenarios. Theses scenarios are described as follow.

3.4.1. Scenario 1 The scenario 1 is defined as having only one smart sensor connected to the network with parameters reconfiguration. In that way, the slave sensor requests to the master node its initial configuration. The following parameters are passed to the slave node:

- ID: 1;
- Minimum time: 3 s;
- Maximum time: 10 s;
- Compression deviate: 5 m^3/day.

The slave sensor is reconfigured after 3150 seconds. The new configuration of the slave sensor is visualized as follow:

- Minimum time: 5 s;
- Maximum time: 20 s;
- Compression deviate: 9 m^3/day.

3.4.2. Scenario 2 The scenario 2 defines the execution of two slave sensor, in which the second slave sensor is connected 2000 seconds later than first slave sensor. Slave sensor 1 is configured with the parameters as follow:

- ID: 1;
- Minimum time: 3 s;
- Maximum time: 10 s;
- Compression deviate: 5 m^3/day.

Slave sensor 2 is configured with the parameters as follow:

- ID: 2;
- Minimum time: 5 s;
- Maximum time: 20 s;
- Compression deviate: 5 m^3/day.

The master node is responsible by storing the graph generation sensor data with the relevant data from each smart sensor. The smart sensor 1 uses the data stored in database 1 while smart sensor 2 uses the data stored in database 2.

3.5 Results

This section presents the obtained results from all test scenarios described above. Figures 7 and 8 show the raw data from the sensor 1 and sensor 2, respectively.

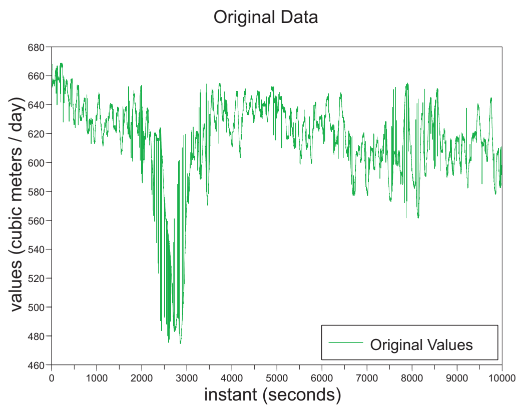

Fig. 7. Original Data from the Slave Sensor 1.

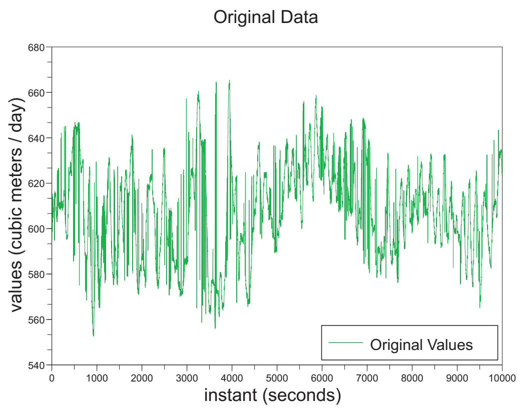

Fig. 8. Original Data from the Slave Sensor 2.

3.5.1. Scenario 1 In this scenario was defined the operation of only one slave with reconfiguration during its activity. The supervision system requests the reconfiguration of the slave node compression parameters 3150 seconds after its initiation. For this scenario, it is calculated the compression rates and the mean square errors for all sensor activity and before sensor reconfiguration and after sensor reconfiguration activities.

Figure 9 shows the slave sensor relevant data. For this scenario the compression of all sensor activity was 90.76% with a mean square error of 44.25. The compression rate before the reconfiguration was of 85.52% with a mean square error of 86.76. After the slave sensor reconfiguration, the compression rate was of 93.16% with a mean square error of 24.71.

3.5.2. Scenario 2 In this scenario was defined the operation of two slave smart sensor. The slave

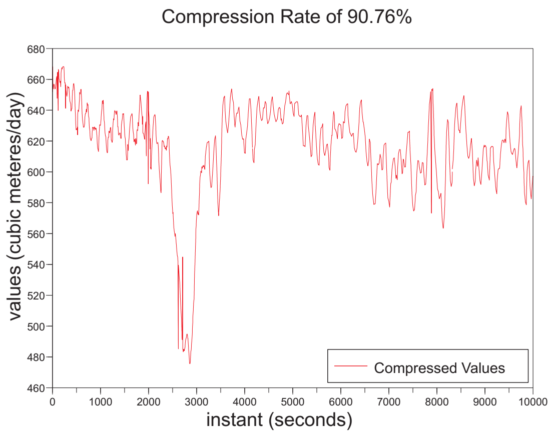

Fig. 9. Relevant Data of Slave Sensor 1 with reconfiguration.

sensor 2 was connected 2000 seconds after the slave sensor 1. For this scenario, it is calculated the compression rate and the mean square error for each slave sensor. The compression rate for slave sensor 1 was 85.80% with the mean square error of 41.13. The compression rate for slave sensor 2 was 90.16% with the mean square error of 32.58. Figure 10 and 11 shows the relevant data of the 1 and 2 slave sensor, respectively.

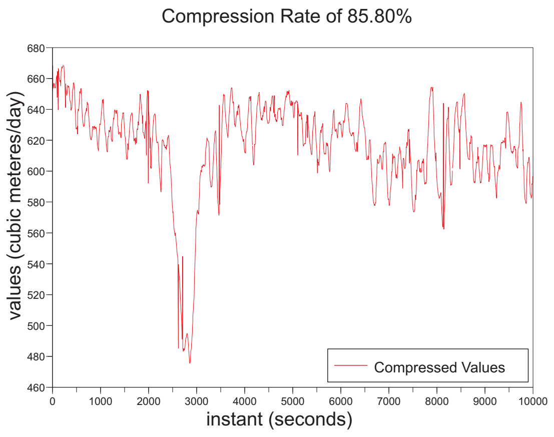

Fig. 10. Relevant Data of Slave Sensor 1.

4. CONCLUSION

The paper presented an event-triggered smart sensor network architecture and its implementation based IEEE 802.15.4 standard. The main feature of this architecture is the integration of a data compression algorithm with simple implementation into the smart sensors. This approach saves network bandwidth, because the sensors only send relevant data.

With the achieved results through the realized experiments, it is possible to conclude that the local compression process in each smart sensor can

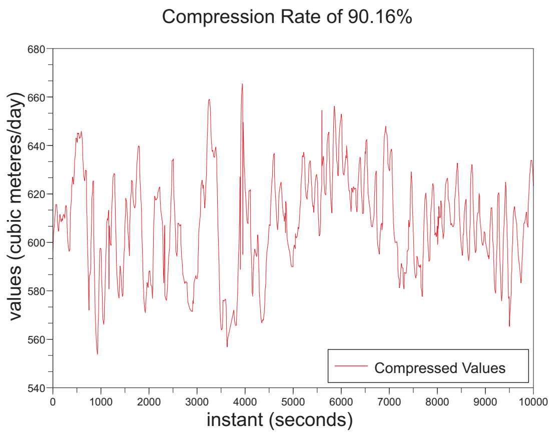

Fig. 11. Relevant Data of Slave Sensor 2.

decrease considerably the data exchanges in the communication network.

REFERENCES

Baronti, Paolo, Prashant Pillai, Vince Chook, Stefano Chessa, Alberto Gotta and Y. Fun Hu (2007). Wireless Sensor Networks: A Survey on the state of the art and the 802.15.4 and ZigBee Standards. *Computer Communications* **30**, 1655–1695.

Bristol, E. H. (1990). Swinging Door Trending: adaptive trend recording?. In: *ISA National Conf. Proc.*. pp. 749–753.

Dolejs, Ondrej, Petr Smolik and Zdenek Hanzalek (2004). On the Ethernet Use for Real-Time Publish-Subscribe Based Applications. In: *Proceedings of IEEE International Workshop on Factory Communication Systems*. pp. 39–44.

Elmenreich, Wilfried (2006). Time-Triggered Smart Transducer Networks. *IEEE Transactions on Industrial Informatics* **2**, 192–199.

IEEE 802.15.4 Standard (2006). Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Specification.

Kopetz, H. and TU Wien (2003). OMG Smart Transducer Specification II. Specification.

Leão, Erico Meneses, Luiz Affonso Guedes and Francisco Vasques (2007). An Event-Triggered Smart Sensor Network Architecture. In: *Proceedings of 5th IEEE International Conference on Industrial Informatics, INDIN 2007, Vienna, Austria.*

Ocera (2002). Wp2 - architecture specification. deliverable d2.1 - architecture and components integration.

Ramamritham, Krithi (1993). Real-Time Databases. *International Journal of Distributed and Parallel Databases* **1**, 199–226.