Reconfigurable Industrial Process Monitoring using the CHROMOSOME Middleware

Stephan Sommer*, Michael Geisinger†, Christian Buckl†, Gerd Bauer‡, Alois Knoll*

*Technische Universität München, Boltzmannstraße 3, D-85748 Garching, Germany, {sommerst,knoll}@in.tum.de
†fortiss GmbH, Guerickestraße 25, D-80805 München, Germany, {geisinger,buckl}@fortiss.org
‡efm-systems GmbH, Reinsburgstraße 96/1, D-70197 Stuttgart, Germany, bauer@efm-systems.de

***Abstract*—In all areas of industrial production and construction, companies have to implement processes with increasing complexity and reliability to satisfy their customer's needs with respect to product quality and continuity. Fulfilling these requirements means to employ continuous process monitoring and documentation without increasing production cost significantly even in case of subsequent reorganization of the production facility. The Multifunk research project focuses on the development and integration of features into the CHROMOSOME middleware to provide the basis for an intelligent, self-adapting process monitoring framework for production quality assurance. The key features of the framework are self-organization and self-configuration based on meta information in combination with industry standard protocols to guarantee an easy integration into industrial processes. The CHROMOSOME middleware is used on top of smart sensor platform MST2012 that has been developed in the course of the research project to tie established analog sensors to the digital process monitoring world. The contribution of this paper is a flexible, self-adaptive embedded middleware in combination with a smart sensor gateway for industrial process monitoring.**

I. Introduction

In the past decades, the complexity of industrial production processes has increased. Nowadays, it is often not only important for a produced good to pass quality checks at the end of the production process, but the individual production steps as well as the collected data need to be documented and stored for later reference. These *monitoring* systems can be quite complex themselves. Hence, the motivation of the Multifunk[1] research project was to ease deployment and configuration of sensor networks for process monitoring such that the complexity visible to the developer is low. This is achieved by two major concepts: on the one hand, a model-driven development tool that is used for code generation abstracts from the underlying complexity of the system and provides the developer with a functional and topological view of the system. On the other hand, self-adaptation and self-configuration features are built into the communication infrastructure such that changes in the sensor network topology are automatically detected and reacted upon. The latter point is of special importance considering that production facilities are nowadays often rearranged and reconfigured to match changed production requirements.

Process monitoring is concerned with the recording of data from the production process and, among others, storing the data for later reference. In many production processes, the same data are used to control the actual process. This is why analog sensors are directly connected to *Programmable Logic Controllers* (PLC) to obtain the data required for process control *and* monitoring in classic industrial control and automation scenarios. The obtained data is then periodically forwarded to a database for permanent storage. However, this setup has two major drawbacks:

1) The described setup makes the monitoring process very inflexible, because the collection of monitoring data is tightly integrated into the control programs of the PLCs that are responsible for the actual production and changes in either part of the program will affect the other.
2) The collection of data for process monitoring is not implemented independently from the sensing required for process control itself. Hence, if the single faulty sensor provides values that "look good", but do not actually represent the physical state of the plant, both the control as well as the monitoring will not ring an alarm bell, although the product will be faulty.

To get an independent instance for the production documentation, an individual sensor network only responsible for the documentation is reasonable. To decrease cost for the additional sensor network, integrating sensors and electronics of both sensor networks into the same physical housing is a cost efficient tradeoff as long as the sub-systems stay independent of each other (with respect to sensing). This also allows placing sensors for process monitoring at optimal locations without directly influencing process control. It also allows adaption of the sensor network without touching the PLC programs responsible for process control, which is crucial in a running production plant.

In our paper, we present a setup consisting of hardware (MST2012 smart sensor system) and software (CHROMOSOME middleware [1]). MST2012 is equipped with two independent data sampling and processing channels to provide independent data for process control as well as for process monitoring. The data transport and validation is done with the CHROMOSOME middleware which provides us with self-adaptation based on a meta information driven *publish and subscribe* paradigm and health monitoring for the system.

The remainder of this paper is structured as follows: the related work and a system architecture for sensor networks in control applications are described in Section II. A description of the CHROMOSOME middleware and its key components is presented in Section III. The real world application scenario and the MST2012 smart sensor are discussed in Section IV.

[1] http://www.multi-funk.de/

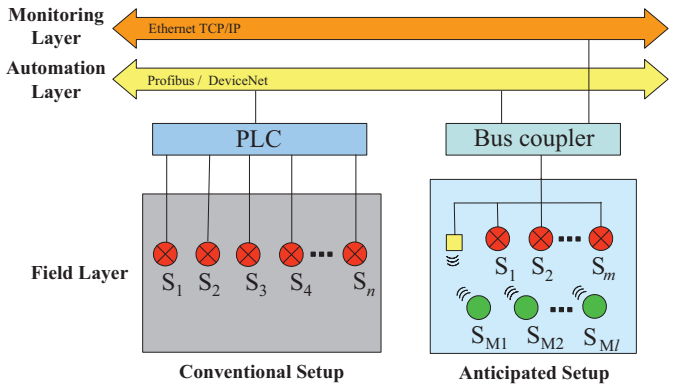

Fig. 1. Exemplary monitoring system setups (conventional and anticipated).

Finally, the paper is concluded with a summary of our experience and directions for future work in Section V.

II. Related Work

A classical design pattern for control applications in industry is to directly connect sensors to a PLC. However, this prevents system designers from using these sensors in more than one control task. To cope with changing surrounding conditions, flexible systems are required to allow adaptations and extensions to comply with future requirements. An overview of requirements and state of the art in industrial sensor networks (wired and wireless) is given in [2]. Key features for industrial scenarios are the integration of already deployed sensors into new applications with reasonable effort as well as the integration of newly deployed sensors into already running applications and the seamless interconnection to business applications.

Both monitoring scenarios are depicted in Figure 1: in the automation domain, deployments are typically divided into three layers. The lowest layer is the *field layer* where wired (S_i) and mobile (S_{Mi}) sensors are located and local control tasks are performed using PLCs. Communication between PLCs is realized in the *automation layer*. Finally, the *monitoring layer* provides interfaces to business applications like process planning and management tools.

The bottom left of Figure 1 shows a common automation scenario with sensors directly connected to a PLC, whereas the sensors on the right are connected to an industrial bus using bus couplers. Unlike the design on the left, the design on the right allows to use sensors in multiple control tasks. To integrate different sensors from different vendors into multiple applications, at least a standardized interface to the sensors like IEEE 1451 [3] is required as a basis. As soon as sensors are connected using a standardized interface, access to this layer needs to granted to the next layer by, for example, a uniform sensor network gateway providing the sensor data to consumers located at the *monitoring layer*. Using this approach, data acquisition can be treated as a service provided by the sensor network. The feasibility of a *service oriented approach* (SOA) for sensor networks is discussed in [4], [5].

The data centric communication paradigm in combination with an optimized and flexible middleware can cope with the mentioned challenges. One well known implementation of such a scheme is provided by the OMG Data Distribution Service (DDS) [6]. In contrast to CHROMOSOME, DDS does not support meta information and seems not suitable for resource constrained systems. Although the requirement to support meta information for a dynamic, self-adapting system consisting of many independent and decoupled parts is already known to be a key factor for years [7], self-adaptive middleware implementations for embedded systems using this technique are still rare. Summing up the related work, there are solutions covering a subset of the requirements in the domain, but there is none covering all of them by itself.

III. CHROMOSOME Middleware

CHROMOSOME [1] is a middleware and runtime system targeting, among others, distributed embedded systems and their connection to PCs. It is common knowledge that a high amount of the code developed for networked embedded systems is infrastructure code and does not depend on a specific application. Testing the code is at least as time consuming than writing it [8], and hence parts of the system which include high complexity should be reused if possible. CHROMOSOME implements a *publish and subscribe* communication paradigm that is based on data centric communication as well as a platform independent runtime environment and hence allows a developer to concentrate on his application rather than the infrastructure. It is designed in a modular way and can be configured according to the intended use case.

Due to its discovery mechanisms, CHROMOSOME is applicable to dynamic networks, where data producers and consumers change over time. Finally, CHROMOSOME is open source and free of charge, which provides a good basis for long term availability. The key-features that make CHROMOSOME a good choice for this application are:

- Support for resource constrained embedded systems.
- Decoupling of senders and receivers of data due to data centric communication paradigm.
- Flexibility and self-adaptability.
- Reliability due to health monitoring.

In this section, we will stress the self-adaptability concepts of CHROMOSOME and introduce the respective architecture and software components (compare Figure 2).

A. Architecture

Figure 2 shows the structure of a single node that runs the CHROMOSOME middleware. The very bottom shows an excerpt of the *hardware periphery* in gray color. The next layer is the *hardware abstraction layer* (green color), which provides a uniform interface for the remaining part of the middleware and ensures that applications can be easily ported to different target platforms. In the case of MST2012, we are using the FreeRTOS [9] operating system on the ARM target and Windows for the PC counterpart. Based on this foundation layer, the *primitive components* (violet) and *core components* (yellow and orange) are built. Core components provide the basic system functionality like communication and scheduling while primitive components provide services for hardware access arbitration. Depending on the platform, the

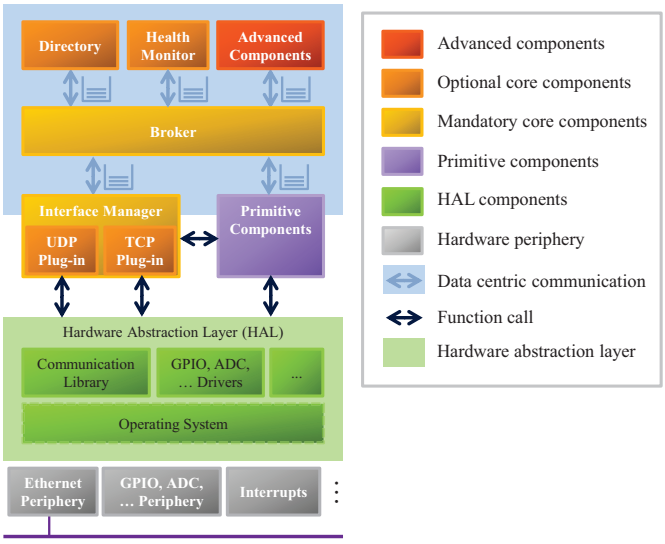

Fig. 2. Simplified diagram of CHROMOSOME node system architecture.

number of core components differs but a fixed set of mandatory components (yellow) is always present. One example is communication: depending on the required communication characteristics, the user can select between UDP and TCP traffic as well as between unicast and multicast messages. By default, communication is achieved using multicast to lower the network load. Finally, application-specific *advanced components* (shown in red color) implement the actual application behavior based on the software components from lower levels. This also includes the components implementing the logic for the MST2012 sensor gateway.

In the following paragraphs, a subset of components will be discussed in more detail, namely *Directory* and *Health Monitor*.

B. Directory Component and Data Centric Communication

In CHROMOSOME, data flow between software components is not specified explicitly. Instead, publishing and subscribing components specify the type of data they produce (so-called *topics*) as well as *meta information* (e.g., timstamp of current sensor value, tolerance of physical sensor). This communication pattern is called *data centric communication*. The information about publications and subscriptions are forwarded to the *directory* component, which is currently deployed to one dedicated node in the network only. The *directory* stores a representation of the subscription and publication requests as well as the attached meta-data. Whenever a new request is issued or a change in network topology requires reconfiguration, the *directory* matches publisher and subscriber information against each other and establishes logical routes within the network based on topology information that is preconfigured or obtain by neighborhood detection during runtime. In the latter case, logical route calculation is an optimization problem with parameters such as resource and bandwidth usage. Establishing of logical routes is performed by adapting the routing information in all nodes along the communication path. These updates are sent to the respective nodes via dedicated route management data channels that are statically allocated between the *directory* and each node.

Once a data path has been set up, the *broker* component, which is available on every node, forwards the data locally to the destination (for incoming data) or the respective network interface (for outgoing data or data that use the respective node as a gateway for multiple networks).

In case this dynamic behavior is not required or intended, a subset or all data routes between components can also be pre-configured and established statically to save memory and processing power on small scale devices.

C. Health Monitor Component

A health monitoring concept ensures proper behavior of the distributed system. This feature is implemented in the *health monitor*, a local component on every node, and a *health manager* component which is deployed on a dedicated node. Depending on the monitoring requirements, it can be sufficient to monitor if data is sent regularly by all networked nodes (alive-signal). In addition, regular checks of the hardware (using predefined test cases) can be scheduled.

For our application scenario, it is sufficient to know that all nodes are alive and, in addition, to know that the gathered data is within a certain range. These requirements are implemented in form of runtime tests in the *health monitor*. The test results are recorded to a log file and, in addition, reported to the user.

IV. Use Case

The MST2012 sensor system has been initially developed for a plant in which unfinished tires are pressed under high temperature to form the final tires. During processing, temperature and pressure profiles have to be precisely met to get a high quality tire which is safe to use for many years. Different tire models have different temperature/pressure profiles. Hence, temperature profiles should be acquired and stored as quality demonstration record.

As explained in the introduction, it is meaningful to separate acquisition of sensor data for process control (here the PLCs that control the tire press) from the sensor data for quality assessment (here a separate sensor network attached to a PC with a database system). Using this arrangement of sensors allows checking the quality of each sensor reading by comparing the values of the redundant sensors. Hence, broken sensors, sensor drift and probably even fouling (accumulation of unwanted material at the sensor) can be detected. In an additional process, the data recorded by the analog sensors connected to the PLCs is also compared to the process monitoring data.

Redundant acquisition of sensor data is one of the requirements for the setup. Furthermore, the monitoring system should be easily reconfigurable and self-adaptable such that subsequent changes to the processing equipment (e.g., for the production of a new type of tire) should not be blocked or deferred by the monitoring system.

The MST2012 smart sensor gateway (compare Figure 3) is introduced to implement the redundant acquisition of sensor data and to meet the requirements for self-adaptability. It consists of two independent processing paths, where each path is connected to a dedicated high-resolution analog to digital

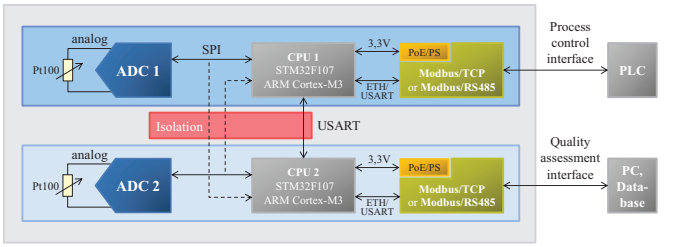

Fig. 3. Schematic diagram of MST2012 sensor gateway: the upper (process control) and lower (quality assessment) paths in MST2012 are isolated from each other to minimize influence of quality assessment on the (time-critical) process under control. On Ethernet, Power over Ethernet (PoE) can be used; otherwise a separate power connector and power supply module (PS) is used. Each CPU uses its dedicated ADC only by default.

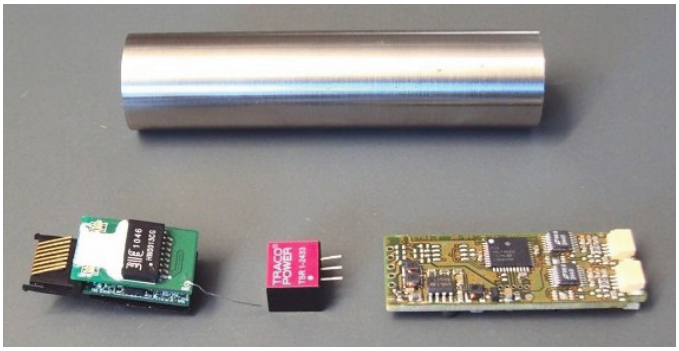

Fig. 4. MST2012 prototype with Ethernet, DC/DC and CPU+ADC board.

converter (ADC) using the SPI bus. In this scenario, two Pt100 temperature sensors are connected to the ADCs.

Data processing occurs independently on each microcontroller and the result is provided to the consumers by a RS485 [10] or Ethernet interface. The respective part of the sensor system might also be powered using Power over Ethernet (PoE) [11], reducing the number of cables involved compared to conventional digital sensor systems. For compatibility reasons, the protocol Modbus, which is widely used in industry and supported by many PLCs, is used as communication protocol (either Modbus/RS485 or Modbus/TCP).

V. Conclusion

In this paper the demand on smart and reliable sensor infrastructure for industrial process monitoring is illustrated and a solution, the smart sensor gateway MST2012 in combination with the CHROMOSOME middleware is provided. MST2012 replaces analog sensors for temperature and pressure measurements. It is designed for industry compliance with respect to field bus support and use in harsh environments.

"Smart" sensors have the capability of running applications locally on the sensor nodes. This also makes the whole system much more flexible, because self-adaptation functionality is implemented directly at the *field layer* (compare Figure 1). Due to the two-channel design, the reconfiguration of the process monitoring system can take place without disruption of the process control system.

To cope with the challenges like reliability, safety and heterogeneity, key features from the data centric middleware CHROMOSOME are used and extended, such as self-organization of the network.

Our experience using the data centric approach shows that this development pattern can reduce overhead and development time for future applications due to its separation of concerns. The developer can focus on his tasks and does not have to take care about the infrastructure.

The whole project is carried out in close cooperation with our partners form industry to ensure that the results are applicable in practice. As soon as the final development steps have been completed to make the design industry compliant (e.g., IP67 compliant housing as depicted in Figure 4) MST2012 and the CHROMOSOME middleware will we deployed to the real industrial setting for a long term test.

As future work, we plan to introduce security features to guarantee a trustworthily communication between devices as well as scalability tests in real world settings. We are also working on a graphical deployment and management tool based on the model driven approach and will be used to create initial configurations and template code using a code generator.

Acknowledgments

We would like to thank the German Ministry of Education and Research (BMBF) for funding the project under grant number 16SV3883 and the VDI/VDE Innovation + Technik GmbH for project supervision. In addition, we would like to thank our partners, especially T.V.P. Gerds GbR[2], for their input and the many fruitful discussions.

References

[1] "CHROMOSOME Middleware," fortiss GmbH. [Online]. Available: http://chromosome.fortiss.org/

[2] A. Flammini, P. Ferrari, D. Marioli, E. Sisinni, and A. Taroni, "Wired and wireless sensor networks for industrial applications," *Microelectronics Journal*, vol. 40, no. 9, pp. 1322–1336, 2009.

[3] D. Wobschall, "IEEE 1451—a universal transducer protocol standard," in *Autotestcon, 2007 IEEE*, Sep. 2007, pp. 359–363.

[4] A. Scholz, C. Buckl, S. Sommer, A. Kemper, A. Knoll, J. Heuer, and A. Schmitt, "eSOA - service oriented architectures adapted for embedded networks," in *Proceedings of the 7th International Conference on Industrial Informatics*, Jun. 2009.

[5] F. Golatowski, J. Blumenthal, M. Handy, M. Haase, H. Burchardt, and D. Timmermann, "Service-oriented software architecture for sensor networks," in *Proceedings of the International Workshop on Mobile Computing (IMC'03)*, Rockstock, Germany, 2003.

[6] G. Pardo-Castellote, "OMG data-distribution service: Architectural overview," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*. IEEE, 2003, pp. 200–206.

[7] F. Curbera and N. Mukhi, "Metadata-driven middleware for web services," in *Proc. of the 4th International Conference on Web Information Systems Engineering (WISE 2003)*. IEEE, 2003, pp. 278–283.

[8] V. Encontre, "Testing embedded systems: Do you have the guts for it," *IBM, November*, 2003.

[9] R. Barry, "Real time application design using FreeRTOS in small embedded systems," 2003.

[10] B&B Electronics Mfg. Co. Inc., "RS-422 and RS-485 appl. note," 2006.

[11] IEEE Computer Society, "IEEE 802.3at-2009 Part 3: carrier sense multiple access with collision detection access method and physical layer specifications - Amendment 3: data terminal equipment power via the media dependent interface enhancements," 2002.

[2] http://www.tvp-online.de/