Customizing Sensor Nodes and Software for Individual Pervasive Health Applications

Peter Langendoerfer, Frank Vater and Krzysztof Piotrowski
IHP,
Im Technologiepark 25,
15236 Frankfurt (Oder), Germany,
Email: {langendoerfer|vater|piotrowski}@ihp-microelectronics.com

***Abstract*—in this paper we introduce the idea of realizing pervasive health care applications by tailoring the underlying hardware and software to the needs of this specific application. We illustrate the feasibility of our idea by discussing a generalized approach for combining different hardware components as well as a generalized approach for adapting specific software components.**

Keywords: pervasive healt care, sensor nodes, 802.15.4 configurable sensor systems, hardware-component, software components

I. Introduction

Wireless Sensor Networks (WSN) can be seen as the key enabler for pervasive healthcare. They provide low cost networked devices which have some sensing capabilities included. Even if the form factor of state of the art sensor nodes is still not unobtrusive to be implantable, it is feasible to use WSNs to monitor vital data in Body Area Networks (BAN). To become widely accepted, sensor networks need to provide:

- Long up times, i.e. need to be energy efficient
- Proper security/privacy features
- Ease of use

These features are still attracting serious research effort, in other words, satisfying solutions are not yet developed. This holds true even for somewhat specific applications such as environmental monitoring [10]. The situation becomes even more complex if the application area covers a broad range. In the field of pervasive healthcare the requirements towards the sensor nodes reaches from stationary devices used for monitoring issues in a flat to tiny battery powered sensor nodes used for vital parameter monitoring.

In this paper we discuss the idea of a Lego like approach to realize customized wireless sensor network systems. A WSN system consists of the hardware platform and the software running on that platform. We are convinced that the development of pervasive healthcare systems is significantly improved if the requirements of the final system can be taken into account when hardware and software are developed. In order to simplify this task we are proposing to use pre-developed components for hardware and software modules that can be freely combined according to the needs of the application under development.

This work was partially funded by 03IP601 and 01BN0710F

The rest of this paper is structured as follows. First we introduce the philosophy of our approach as well as the hardware and software modules which are currently available in our components library. In section 3 we illustrate how a pervasive healthcare or more correct an ambient assisted living application can be realized on top of a heterogeneous hardware/software platform. The paper closes with a discussion of open issues.

II. The Lego Approach towards customized Sensor Systems

In this section we introduce our idea of building sensor systems analogous to Lego constructions. After the philosophy we describe parts of our toolbox. We start with a subset of the components we have available for realizing customized sensor network systems. This subset is selected according to the components used in the example discussed in the following section. We also indicate the current state of development per component. In addition we introduce two means we currently use to ensure easy integration of components.

A. The Lego Philosophy

It is widely accepted that customized solutions are much better fulfilling application requirements but are more costly compared to general approaches that can also solve a given problem only to some extent. The requirements in the area of pervasive healthcare are cost efficient tiny devices, on one hand and long uptimes, strong protection of user data and extremely high reliability of the system, on the other. They are very difficult to achieve when starting from components off the shelf (COTS). Designing each new system from scratch is putting high development cost and an increased risk of design errors to the development of a new system. It will also negatively affect the time-to-market. Our approach combines the good of using COTS and individual developments. The idea is straight forward and up to certain limit it is analogous to using patterns. We are proposing to develop a set of hardware components ranging from different radio frontends over a set of µController to specialized hardware accelerators for protocols or cryptographic operations. In addition to that, an appropriate toolbox needs to provide also a suitable set of software building blocks such as operating systems, protocol implementations and middleware approaches. To complete the essential ingredients, there is a need for means to combine:

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2009.6014
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2009.6014

- different hardware components with each other,
- different software components with each other,
- software and hardware components,

The most effective and convenient realization of such a Lego like system approach would support a simple drag and drop approach to select and combine modules. I.e. interface adaptation would be done automatically. Also the verifying whether the system under development (SUD) is going to fulfill the requirements or not should be checked automatically during the design phase.

While this full tool support for designing customized health care systems is still future work, we have gained positive experience combining the modules discussed in the following subsections by hand. This experience strongly indicates that our approach will help to improve the design of pervasive healthcare systems. Realizing tool support for this type of design process is still a scientific challenge and some open issues are discussed in section 4.

B. *Hardware Components*

In order to provide a really flexible toolbox a set of µControllers and radio front ends needs to be provided. By that features like ultra low power consumption using appropriate µC or improved transmission range when using an 868MHz based IEEE802.15.4 (V2006) solution can be achieved by selecting the correct components. In addition to that, hardware accelerators for computational intensive operations like cipher means should be provided. In our toolbox we currently provide the following hardware building blocks:

- µC: 32bit MIPS [14], 32 bit Leon V1-3 [4], and an IPMS430 [5, 6] which is instruction set and timing compatible to the TI430 [11]. All these components have already been fabricated successfully.
- Radio Frontends: 802.15.4 (V2006) and 802.15.4a, both front ends are currently under development. For V2006 an FPGA based solution is already working, and the analog part of the 802.15.4a [9] is currently under fabrication.
- Hardware accelerators: we provide AES [7] as secret key means supporting all common operation modi and up to 256 bit key length. As public key mechanism we support elliptic curve cryptography and ASICs for B-233 have been successfully manufactured [8].

C. *Software Components*

In addition to those hardware building blocks we have already implemented/available the following software components:

- Operating Systems (OS): we have used tinyOS [1], Contiki [2], eCos [12] and Reflex [3] in different settings.
- Protocols: we realized 802.15.4 for Reflex, but for other OS such as tinyOS appropriate implementations are available [14].
- tinyDSM [13] is a middleware for WSNs, which provides a reliable distributed storage and an event definition as well as a query language.

D. *Lego Adapters*

Providing interfaces that allow easy and flexible integration of different µC, hardware accelerators and software components is the key enabling part of the whole Lego like approach. If this cannot be ensured by proper means or even better appropriate tool support our approach is collapsing towards a complete redesign for each new system, i.e. nothing is gained. We are convinced that the "silver bullet" for this issue is not available but that some approaches can be generalized in such a way that they can be applied for different hardware and software components, respectively. In the following paragraphs we describe two technologies we successfully used already.

In order to attach hardware accelerators to the µCs available we use standard data bus technologies to exchange data. While this is not a novel approach we decided to attach all types of hardware accelerators to the bus as local memory. Basically the idea is as follows. Data which shall be processed by the hardware accelerator is written into that specific memory area which is assigned to the corresponding hardware accelerator. Then the hardware accelerator starts modifying the input data and after that the processed data can be read back. Listing 1 shows a code fragment which is used to write plain data into the AES "memory". We have used this technique successfully to attach our crypto hardware accelerators to Leon as well as MIPS processors [15].

```
// AES base address
#define AESBASE 0x20200000 //memory mapped IO
//Write key

writeReg(0x2b7e1516, AESBASE + KEYBASE + 0);
writeReg(0x28aed2a6, AESBASE + KEYBASE + 1);
writeReg(0xabf71588, AESBASE + KEYBASE + 2);
writeReg(0x09cf4f3c, AESBASE + KEYBASE + 3);

writeReg(0x3243f6a8, AESBASE + DATABASE + ENCRYPTION + 0);
writeReg(0x885a308d, AESBASE + DATABASE + ENCRYPTION + 1);
writeReg(0x313198a2, AESBASE + DATABASE + ENCRYPTION + 2);
writeReg(0xe0370734, AESBASE + DATABASE + ENCRYPTION + 3);

//Wait for 77 Clock cycles or interrupt
//READ_OUT
```

Listing 1: Memory Like Interface

In the software area we use the concept of separating the core functionality of the software, protocol or middleware or whatsoever from support functions which are OS dependent. Up to a certain extent this approach also solves the issue of generalizing the software component to software component interface adaptation challenge. This holds true for interfaces which, e.g. exchange data via the OS. We have applied this technique successfully to our middleware tinyDSM which is composed into its core functionality and an OS adaptation layer in which memory management functions as well as access to the protocol stack deployed on a sensor node are clustered. We have successfully adapted tinyDSM to tinyOS and Contiki using this approach. Listing 2 shows how tinyDSM core functions are wrapped into tinyOS and Contiki constructs, respectively.

```
//Communication interface process task
//implementation in the tinyDSM core
void CorecommIntProcess_Task(){…}
//scheduling of task (above)
Corepost_commIntProcess();

//------------------------------------------------

//Contiki Wrapper
PROCESS(commIntProcess, "commIntProcess");
PROCESS_THREAD(commIntProcess, ev, data){
  PROCESS_BEGIN();
  CorecommIntProcess_Task();
  PROCESS_END();
}

void Corepost_commIntProcess(){
  process_post(&commIntProcess, 0x81, 0);
}

//------------------------------------------------

//TinyOS Wrapper
task void commIntProcess(){
  CorecommIntProcess_Task();
}

void Corepost_commIntProcess(){
  post commIntProcess();
}
```

Listing 2: OS abstraction layer for tinyDSM

III. Distributed Fall Detection using customized Sensor systems

In this section we shortly describe the architecture of healthcare application which provides monitoring vital parameters and fall detection, see Figure 1. The idea is to monitor and record vital parameters via a BAN, and to monitor the activity of a person via sensor nodes installed at fix points in the flat. The fall detection is realized by using an accelerometer in the BAN which indicates that a person is moving extremely fast, i.e. is falling if it measures g values equal to or greater than g. If such a value is detected the fixed sensor nodes start the positioning algorithm. The distribution of the measurement data and the definition of the application logic, i.e. how the system reacts on which event is done using the tinyDSM middleware.

The application sketched here can be realized by applying the novel approach introduced here. All needed components are available or currently under development. The system we envision consists of two types of sensor nodes. The first type is used to set up a BAN. The major task of these sensor nodes is to record vital parameters of a person. There is no computation intense operation to be carried out by these nodes so they are configured with a 16-bit µC. The secrecy of the personal data is ensured by encrypting it using the AES hardware accelerator. These sensor nodes support at least one SPI interface via which sensing devices, e.g. to detect heart rate, can be attached to the sensor node. Figure 2 shows the sensor node configuration.

The second type of sensor nodes is installed at fixed points in the flat which may alleviate energy issues. These sensor nodes are used to track the position of the persons moving around in the flat and can be used as gateway nodes to a care taking agency. Since these nodes are supposed to calculate the position of the person living in the flat they are equipped with a 32-bit µC, here a Leon. In order to support the positioning as part of the pervasive healthcare application both types of sensor nodes are equipped with pulse based ultra wide band (UWB) 802.15.4a radio, which enables fine granular positioning. Figure 3 depicts this architecture.

The hardware accelerators for cryptographic operations, i.e. AES and ECC have been integrated in both architectures using the memory like interface discussed in section 3.

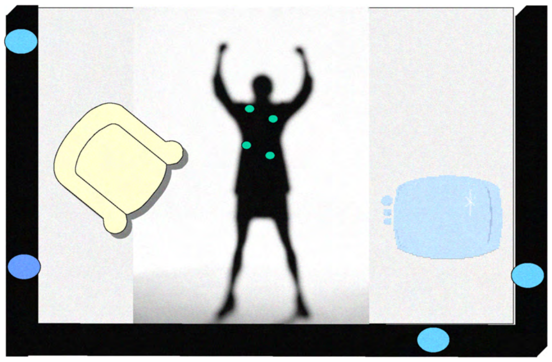

Figure 1: Pervasive health care application: monitoring of vital parameters vis BAN (green dots) plus fall detection via fix installed sensor nodes (blue dots)

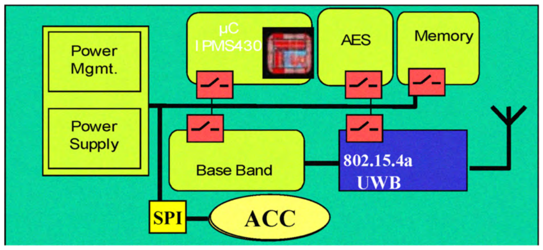

Figure 2: Sensor node for BAN based monitoring of vital data

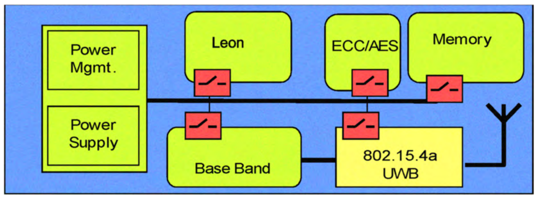

Figure 3: Sensor node for motion monitoring and positioning

The software architectures of the two types of sensor nodes are different as well, see Figure 4. The BAN sensor nodes use Reflex as OS, since it comes with an extreme small footprint. On the sensor nodes used for the motion tracking and positioning the eCos is used as OS, due to the fact that it was developed for Leon processor, which is integrated in those sensor nodes. The cooperation between the two different systems is achieved by employing a standard protocol, i.e. 802.15.4a and by using a common middleware, i.e. tinyDSM. Both software components have been adapted to the operating systems. For tinyDSM this was achieved via the above mentioned OS adaptation layer.

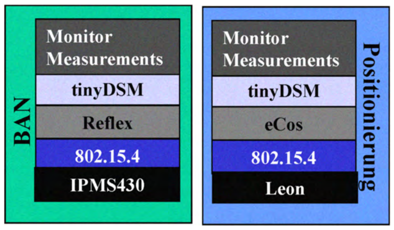

Figure 4: Software Configuration of the BAN and the fixed sensor nodes

The cooperation of the two types of nodes is achieved by defining the following events:

```
1.    Event ACC IF acc >= 1g trigger update()
2.    Event ACC IF acc >= 1g trigger
      positioning();
```

The first event is defined on the BAN node which is attached to the accelerometer. It immediately sends messages with the new measurements to all sensor nodes in the system. The second event is defined on the positioning sensor nodes and starts a new positioning if the new acceleration value is equal to or larger than one, which might indicate that the person is fallen. If the position of the BAN sensor nodes is retrieved to be at the floor level an alarm message can be sent to the care taking agency.

IV. CONCLUSIONS

In this paper we have introduced the concept of configurable sensor systems, i.e. software and hardware which is composing a single sensor node but also the complete WSN. We have discussed two techniques we successfully applied for combing hardware block, i.e. the memory like interface, and the OS adaptation layer which we used to adapt our middleware platform tinyDSM in an easy and efficient way to different operating systems.

While the results we achieved so far are very promising a lot of research challenges still needs to be tackled. One open issue is how the availability of hardware accelerators can be propagated to high abstraction layers or be integrated into customized compilers. Another open issue is how tools can support the proper configuration of the overall system, and how interfaces can be selected or even better adapted automatically. We will address these issues in our future work.

REFERENCES

[1] J. Hill, P. Levis, S. Madden, A. Woo, J. Polastre, C. Whitehouse, R. Szewczyk, C. Sharp, D. Gay, M. Welsh, D. Culler and E. Brewer, TinyOS Homepage: http://www.tinyos.net

[2] A. Dunkels, B. Gronvall and T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors", In First IEEE Workshop on Embedded Networked Sensors, 2004

[3] BTU Cottbus, REFLEX - Realtime Event FLow EXecutive Homepage: http://idun.informatik.tu-cottbus.de/reflex/

[4] Z. Stamenković, C. Wolf, G. Schoof and J. Gaisler "LEON-2: General Purpose Processor for a Wireless Engine " , DDECS 2006

[5] IPMS430 Processor User's Manual, IPMS Fraunhofer, Dresden 2007.

[6] IPMS430 Processor VHDL Model, IPMS Fraunhofer, Dresden 2007.

[7] F. Vater, P. Langendörfer, "An Area Efficient Realization of AES for Wireless Devices", it - Information Technology, Volume 49, Issue 3/2007, Pages 188-193, doi 10.1524/itit.2007.49.3.188

[8] St. Peter, P. Langendörfer, "An Efficient Polynomial Multiplier in GF(2m) and its Application to ECC Designs", in Proceedings of Design Automation and Test in Europe (DATE) Conference, IEEE Society Press, 2007

[9] G.Fischer, O. Kleymenko, „UWB radio transceiver implementation and measurement results"ICUWB 2008, Sept. 2008, Hannover, Germany

[10] K. Rerkrai, C. Jardak, A. Kovacevic, J. Riihijärvi and P. Mähönen "Survivable and Scalable WSN Solution for Environmental Monitoring in Harsh Conditions" EWSN 2009, Demo program abstract, Cork, Ireland, February 2009

[11] MSP430x1xx Family User's Guide, Texas Instruments, Dallas 2006.

[12] eCos (embedded configurable operating system) http://ecos.sourceware.org/

[13] K. Piotrowski, P. Langendoerfer and St. Peter, "tinyDSM: A Highly Reliable Cooperative Data Storage For Wireless Sensor Networks", accepted for the 2nd International Workshop on Distributed Collaborative Sensors Networks (DCSN09), to appear in the Proceedings of of the 2009 International Symposium on Collaborative Technologies and Systems (CTS09)

[14] An IEEE 802.15.4 protocol implementation (in nesC/TinyOS): Reference Guide v1.0 http://www.hurray.isep.ipp.pt/asp/show_doc2.asp?id=300

[15] St. Peter, M. Zessack, F. Vater, G. Panic, H. Frankenfeldt and M. Methfessel, "An Encryption-Enabled Network Protocol Accelerator", in Proceedings of the 6th International Conference on Wired/Wireless Internet Communications (WWIC 2008), May 28-30, 2008, Tampere, Finland