A Platform for Lab Exercises in Sensor Networks

Thomas Fuhrmann[1] and Till Harbaum[2]

[1] IBDS System Architecture, Universität Karlsruhe (TH), Germany
[2] Beecon GmbH, Haid- und Neu-Str. 7, 76131 Karlsruhe, Germany
fuhrmann@ira.uka.de, harbaum@beecon.de

Abstract. Programming of and experiences with sensor network nodes are about to enter the curricula of technical universities. Often however, practical obstacles complicate the implementation of a didactic concept. In this paper we present our approach that uses a Java virtual machine to decouple experiments with algorithm and protocol concepts from the odds of embedded system programming. This concept enables students to load Java classes via an SD-card into a sensor node. An LC display provides detailed information if the program aborts due to bugs.

1 Introduction

Sensor networks are about to leave the forefront of research in distributed systems and become a standard topic that is taught both in lectures and laboratory exercises. Those of us who have been promoting research in sensor networks for the last years are well acquainted with the various devices that are used for sensor network prototypes. But those whose core skills are outside the field of low-power embedded devices often struggle with the odds that come with such tiny devices.

About a year ago, while preparing a lab-course on telematics at the University of Lübeck (Germany) we have begun to develop a new approach that makes experiments with sensor networks more easy. This workshop presentation reports on our experiences with that platform. Its focus lies hence more on the didactic aspects and their practical realization than on the study of novel aspects in sensor networks in general.

Sensor network platforms for research are abundant [1][2][3]. Typically, these platforms need to be time consumingly flashed with a code monolith consisting of low-level operating system code, the networking stack, and the respective networked application. This monolithic approach leads to a rather slow development cycle. Moreover, bugs in one part of the code can affect the entire system. And often such systems crash without a direct trace to the cause of the malfunction.

All this renders the typical sensor network platforms ill suited for didactic purposes. Accordingly, we devised an improved platform for our lab-course that is based on our previous *microblue*[1] platform [4], but is easily handled even by unexperienced students. Key ingredient is a small Java virtual machine that implements a subset of the Java 2 standard edition, much like the Java 2 micro edition does [5]. The board (cf. fig 1) provides a small 2×16 LC display, four keys, and an SD-card slot. It can be equipped

[1] The microblue stack is commercially sold by Beecon GmbH, a spin-off of the Karlsruhe university. Microblue is used by customers both in academia and industry.

with an 868 MHz radio transceiver, a Bluetooth module, or up to two RS232 compliant sub-D connectors.

The following sections describe the didactic rationale behind our design (sec 2), the architectural features that were implemented so far (sec 3), and first experiences with our platform (sec 4).

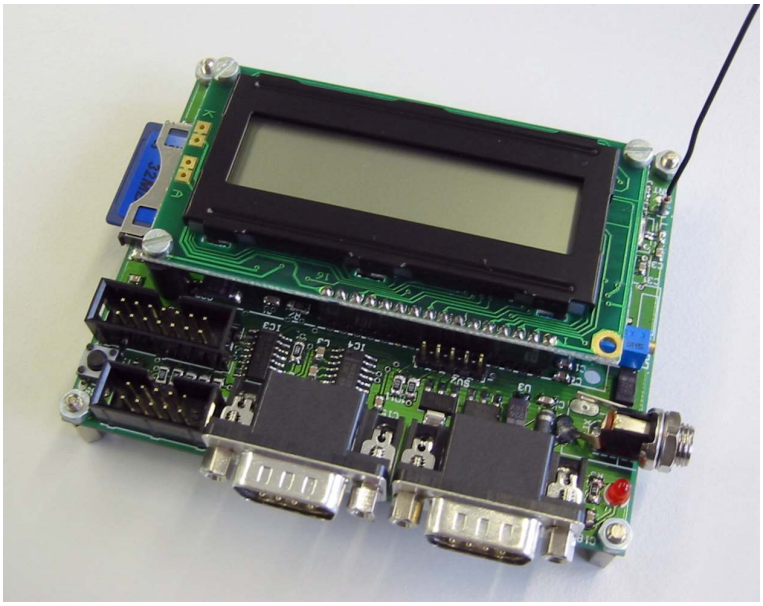

Fig. 1. The sensor network board used in the lab-course at the University of Lübeck

2 Design for Didactics

The lab-course's ancestor was a summer school taught by one of the authors in 2000. There, the students had to design and implement an entire protocol stack in Java that used the serial ports of desktop PCs to create a functional toy network. This course was refined over the years.

The didactic concept of the course is to divide the material into small units, each of which can be handled by one student. Typical units are e.g. an error detection method, an access mechanism to a shared medium, or a routing algorithm. The theoretical basis for each unit is presented in a talk by an individual student who also distributes hand-outs. Subsequently the students form teams who implement the different layers of the protocol stack. One goal besides the implementation work itself, is for the students to understand the role of interfaces and to create a clear design of reusable code modules

for the various aspects of the protocols. Finally, the students can compare the different implementations, algorithms and protocols based on actual measurements of their implementations.

This didactic concept created the need to provide a sensor networking platform that supports a modularized software development approach. The students should be able to immediately use a sensor node and substitute individual modules in a given stack with their own implementation as they go along with the course. Especially, should they be able to work with a complete system whose components they can use and replace, without understanding the entire system and without seeing the source code. Errors should be confined and localized as far as possible.

Java has been found to be the best suited language for that course, since many students already had experiences in Java programming, especially with tools like, e.g. JBuilder or Eclipse. Moreover, the fact that exceptions indicated the source code line where they occurred is considered most valuable by the students.

3 Architecture of the Platform

As a consequence of these requirements, we investigated the possibilities to use Java on a small sensor network platform. This has in fact proven feasible. In detail our solution provides the following architectural features:

- By using Java as sole programming means for the user of a node, we were able to separate the node's operating system from the protocol implementation. The Java byte code is inserted into the node via an SD-card. Thus there is no need to flash the microcontroller during the lab-course.
- Exceptions that occur during execution of the Java code are displayed as text message on the node's LC display. If the byte code contains debug information, these messages indicate the source file and the line where the exception originated. Users can scroll through the stack trace if necessary.
- The Java virtual machine (VM) supports multi-threading and thus simplifies many aspects of network programming. All *java.lang* classes that were ported to the node are thread safe.
- Heap memory can be accessed via *new*. Abandoned memory is automatically garbage collected.
- The low level communication interfaces (868 MHz, L2CAP Bluetooth, and RS232 serial) can be reached via simple octet-streams. According to the design goal of the system all protocol aspects have to be implemented in Java.
- Sensor interfaces (e.g. the microcontrollers analog ports) are reached with simple Java wrapper classes.

Altogether, the resulting system is easily used even by unexperienced students. The Java implementation can be coded, compiled and tested on any PC. Then the Java *class* files are simply copied to an SD-card that is inserted into the sensor node. The code is parsed and executed upon reset of the node.

The following section reports on the experiences gained from the implementation and first tests of the new platform.

4 Experiences, Conclusions, and Outlook

Even though the envisaged application of the nodes requires only limited capabilities, this design challenges the limits of our ATmega microcontroller. At 8 MHz the node can on average perform only about 100 000 byte code instructions per second. Heap operations, especially the garbage collection, further consume CPU resources. Nevertheless, this is sufficient for the didactic purpose of the platform.

In order to allow for different (and maybe inefficient) implementations, the board provides 512 KB external RAM, e.g. as heap memory. We currently extend the VM to access this memory via bank switching.

The most pressing problem turned out to be the flash memory. Both the VM and our Bluetooth stack alone easily fit into the 128 KB limit, but we have to engineer hard to fit both into that space at the same time. Currently, we explore modifications of the VM design that we hope can ease that situation.

Applicationwise, we consider our platform a success. We found that the combination of actual sensor node hardware with a simple Java programming approach intensifies the learning experience. Students have a greater freedom than with ready-made equipment. Still there is full freedom for own implementations without the need to get acquainted to the odds of embedded system programming.

References

1. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System Architecture Directions for Networked Sensors. In: Proceedings of the 9th Int. Conf. Architectural Support Programming Languages and Operating Systems (ASPLOS-IX). (2000) 93–104
2. Beigl, M., Gellersen, H.: Smart-Its – An Embedded Platform for Smart Objects. In: Proceedings of the Smart Objects Conference (SOC 2003), Grenoble, France (2003)
3. Beutel, J., Kasten, O., Mattern, F., Römer, K., Siegemund, F., Thiele, L.: Prototyping Wireless Sensor Network Applications with BTnodes. In: Proceedings of the IEEE European Workshop on Wireless Sensor Networks (EWSN) 2004, Berlin, Germany (2004) 323–338
4. Fuhrmann, T., Harbaum, T.: Bluetooth für Kleinstgeräte. In: Proceedings of the Telematik Fachtagung, Siegen, Germany (2003)
5. Topley, K.: J2ME in a Nutshell. O'Reilly (2002)