Poster Abstract: An OO Approach to Sensor Programming

Till Riedel, Andreas Arnold, Christian Decker

***Abstract*—This paper outlines the advantages of applying object oriented modeling principle to sensor network programming. This closes the gap between programming principles in the world of small things and big servers. We show that typing based on classes can make algorithms robust to changing data layouts and implementation changes. Furthermore we present an implementation based on ultra-lightweight java virtual machine running on low-power sensor nodes.**

***Keywords*—Object oriented languages, wireless sensor nodes, system architecture**

I. Introduction

Object oriented programming has become the predominant language used in newly created software. Its virtual machine based runtime makes it easy to port software to a variety of platform without the need of changing implementations. The success of the J2ME (Java 2 Mobile Edition) manifested by millions of cell phone application has shown that OO based programming accelerates development cycles on highly embedded systems Sun introduced with SPOTS (http://www.sunspotworld.com/) platform a 32bit networked embedded system that resembles a sensor node. At the same time the Standard and Enterprise Edition of Java are the programming interface for many commercial business and engineering products.

In order to show how classical ultra-low power 8bit micro-controller based sensor systems can profit from a java virtual machine and can be integrated into a world of object oriented modeling tools we have implemented a Java implementation for Particle Sensor Nodes (http://particle.teco.edu).

The variety of possible sensors and values makes it many difficult cases to interpret the data correctly, for both locally on a senor node and remotely on PC system. In this poster we want to show how object orient modeling and object level helps to build robust data processing algorithm on sensor nodes.

The work presented in this poster was partially funded by the European Community under contract no. 4270 and by the Ministry of Economic Affairs of the Netherlands under contract no. 03060.

Till Riedel is with TecO/Universität Karlsruhe, Vincenz-Prießnitz-Str. 3, 76131 Karlsruhe, Germany (e-mail: riedel@teco.edu).

Andreas Arnold is with TecO/Universität Karlsruhe, Vincenz-Prießnitz-Str. 3, 76131 Karlsruhe, Germany (e-mail: arnold@teco.edu).

Christian Decker is with TecO/Universität Karlsruhe, Vincenz-Prießnitz - Str. 3, 76131 Karlsruhe, Germany (phone: +49 721 464704 15; e-mail: cdecker@teco.edu).

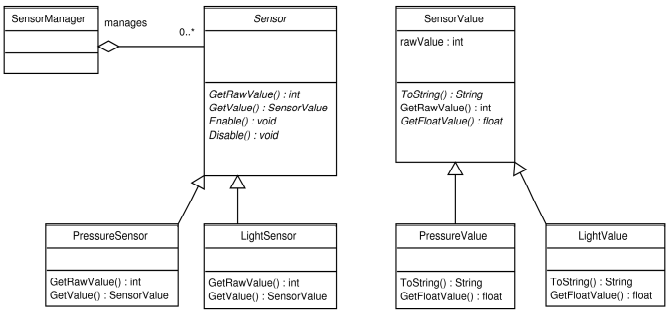

Figure 1: Abstract Sensor Interfaces encapsulation of data

II. Object Oriented Sensors

We follow the vision of classical object oriented (OO) modeling approaches such as [[1]] when we abstract real-world items and data by objects and classes. In the following we want to shortly describe just a few advantages of such an approach for sensor networks.

A. Abstract Sensor Classes

Objects encapsulate and hide the details about an the implementation and data layout. . Many information processing algorithm in sensor network tend to have a binding against the layout of the underlying data. This makes it difficult to reuse code and eventually slows down any development process.

By using abstract sensor classes generic algorithms on e.g. on scalar valued sensor data can be developed and bound to a concrete hardware instance at a late stage, even at runtime. On the most abstract level sensors and sensor values can be used polymorphic as via a very generic interface as shown in Figure 1. The „instance of“ operator and typecasting also allows disambiguation of object classes at a later stage of the processing logic again.

B. Serialization

The data encapsulation especially proves valuable when communicating data. By establishing a common id scheme it is sufficient to serialize the sensor data object together and transmit it to another party in the network. Because the object encoding is fixed inside the class files it is automatically distributed together with the interfaces and the processing logic inside the class. Such a system clearly separates data from layout and code and reduces the typing overhead to a minimum.

C. *Type Safeness*

One of the additional advantages of programming data processing in the Java language over other imperative programming approaches is its type-safeness. On micro-controller based hardware environments this is important, as there is no memory protection enforced in hardware or by the operating system. Type violations in user code can have serious side effects on other systems. Like demonstrated in SPIN project [2], type safe languages allow execution of user code at system privileges. This is why we see another focus of Java in the implementation of a robust operating system like runtime eviroment. A basic set of performance critical subsystems like bus access will still be implemented by compiled machine code. This provides a very thin abstraction layer. Higher abstraction layers providing like sensor sampling and conversion can be loaded dynamically and be executed within the VM. This allows us to design adaptable re-configurable system architecture.

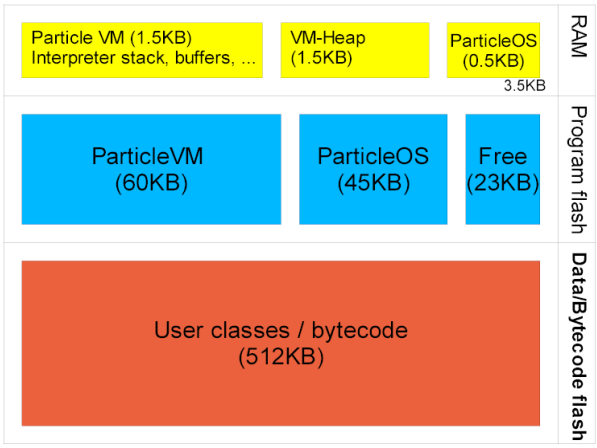

Figure 2: Particle VM Memory Layout

III. Implementation

The implementation of the ParticleVM is based on the lightweight java byte-code interpreter NanoVM [3] and runs on 8bit micro-controllers with a minimum amount of memory. The implementation covers all static features of the Java programming language as it uses code generated by standard java compilers such as the JDK javac (see http://java.sun.com) or jikes (see http://jikes.sf.net). The class files are further optimized on byte-code level, e.g. by elimination of the constant pool.

In our version VM all language features except for exceptions and reflection have been implemented. There is a minimalist java runtime environment for Particle Computers, which maps the basic system functionality provided by the enabling services such as networking and sensory services. The implementation especially covers all performance critical memory, bus and rf access routines.

TABLE I
Initial Benchmark Results

avg. code size	40%
avg. interpretation overhead	3000%

The current 2/32 Particle includes a Microchip PIC18F6720 micro controller. This low power MCU has an instruction cycle of 0,2 µs and includes only 4K of RAM and 128K of code memory. In contrast to directly executed machine code the 512K of external Flash memory can be additionally used as program memory holding Java classes.

The current implementation of the byte code interpreter occupies 60KB of code memory and 1.5K of memory. Additionally 45KB of code memory and 0.5K of RAM are dedicated to the low-level native API for the sensor note with basic operating system features and the RF functionality including message buffers. This leaves 1.5 KB of heap memory that can be used by any user program running on top of the ParticleVM. The complete memory layout is shown in Figure 2.

IV. Performance

The performance characteristics of our first naive implementation are outlined in Table I. Especially code compression and runtime overhead can, however, further be optimized in future version. The numbers were measured benchmarking a simple aggregation functions on sensor values.

The high numbers on the interpretation overhead still show that encapsulated code does always come at higher cost. The question is if at the end of the day the reduction of complexity can even make up this.

V. Conclusions

We showed in this paper that object oriented programming on sensor nodes using java can solve many of the interfacing problems between sensor network applications. A lightweight implementing of a java VM makes it possible to conclude the fully integrate such platforms into a world of backend architectures on language level without proposing large scale middleware solutions. In our ongoing work on this matter want show how sensor networks can be integrated into larger scale application models without loosing the flexibility of a tiny, low power hardware platform.

VI. References

[1] Booch, Grady. *Object-Oriented Analysis and Design with Applications.* Addison-Wesley. ISBN 0-8053-5340-2.

[2] Brian N. Bershad, Stefan Savage, Przemyslaw Pardyak, Emin Gün Sirer, Marc Fiuczynski, David Becker, Susan Eggers, Craig Chambers. *Extensibility, Safety and Performance in the SPIN Operating System,* Proceedings of the 15th Symposium on Operating Systems Principles, pp 267-284, Copper Mountain, Colorado, 1995

[3] Thomas Fuhrmann, Till Harbaum, A Platform for Lab Exercises in Sensor Networks. 4th GI/ITG Fachgespräch Sensornetze Zürich, Schweiz, March 23-24, 2005 pp. 29-32