`dinam`: A Wireless Sensor Network Concept and Platform for Rapid Development

Dawud Gordon, Michael Beigl and Martin Alexander Neumann

***Abstract*—`dinam` is a novel approach to simplified rapid prototyping of wireless sensor network applications as well as an according WSN platform. As opposed to the traditional mote-based development archetype, `dinam` proposes combining the development steps into a single continuous, fluid process that is completely integrated into the node. The `dinam` concept sensor node integrates all development tools, source code and other data into the sensor node system. It is claimed that this concept will greatly reduce the amount of effort required to develop wireless sensor network applications by removing the overhead of installation, iterative development steps and complexity of the development process. In order to confirm or refute this claim, a first prototype for educational purposes is developed and presented which implements the `dinam` approach. The development of applications is evaluated in terms of time required for a specific scenario with a user study. The results presented here indicate that an integrated instruction and development period of 10 minutes is sufficient for simple applications using the `dinam` approach.**

I. Introduction

Development of wireless sensor network applications has historically been a time-consuming and effort-intensive task. Normally, assuming pre-existing hardware, the process involves several components such as sensor nodes, a development environment, libraries, drivers, compiler, linker, assembler, debugger, programming setup (e.g. a hardware programmer device, cable, programming software, or an over-the-air programming (OTAP) approach) and design, documentation and code revision platform. A typical development process works as follows:

1) Design application
2) Set up development environment
3) Connect hardware
4) Prepare libraries and previous code
5) Write code
6) Compile code and link binaries
7) Write binary to hardware
8) Evaluate effects
9) Repeat from 4

Setting up the development environment and connecting it to the hardware devices is time consuming, but must be carried out once at the beginning of the development phase. The remaining steps contain the body of the development process and are repeated many times throughout the course of a project. In most common development platforms, these steps cannot be parallelized and must therefore be carried out sequentially: load/prepare previous code, write code, compile, flash, evaluate.

We refer to the presented method as mote-based design which is typical for a development process for mote-type nodes or embedded systems. Development for mote applications needs to follow a top-down approach in order to avoid costly development delays, which requires careful design and a high degree of expertise. Mote-based development provides good support for experts: different components – e.g. separated libraries – within the development process allow fine-tuning the development process and the functionality of the resulting program. This disciplined top-down analysis and design process is beneficial for complex applications.

However, there is a price to pay for such an approach: due to repeated steps in the development process, development costs for small and simple applications – which are still typical in wireless sensor networks – are high in comparison to the application size. Even worse, the complex interplay of the various components in the development process often leads to incompatibility problems between sensor node hardware versions, libraries, development environment configurations, etc., which are beyond the capabilities of non-expert developers: the number of possible error sources that have to be observed and the range of development components that have to be mastered are high. [7] summarizes that for pervasive sensor network development it is necessary that *"simple things must be simple [to develop]"*.

This overhead can be a hindrance in rapid prototyping in research and education. In research, quick testing of typical key functionality under variable circumstances is obstructed by the rigid, top-down software development approach. In educational settings, students cannot be expected to acquire experience with all the required components, before starting the actual task.

We propose to address these problems with an integrated, interactive, bottom-up development approach: a `dinam` concept sensor node itself contains the program, development environment, program interpreter and debugger as well as all documents in a single device. `dinam` development thus differs from mote development, as number of components and steps of development are considerably reduced.

This paper is structured as follows: we first present related work followed by the introduction of our concept. We then

Manuscript received March 1st, 2010. The authors would like to acknowledge funding by the European Commission for the ICT project CHOSeN Cooperative Hybrid Objects Sensor Networks (Project number 224327, FP7-ICT-2007-2) within the 7th Framework Programme.

D. Gordon and M. Beigl are with the TecO Group at the Karlsruhe Institute of Technology, Karlsruhe (KIT), D-76131 Germany. Email: {dawud.gordon, michael.beigl}@kit.edu

M. A. Neumann is with the Technische Universität Braunschweig, Braunschweig, D-38106 Germany. Email: ma.neumann@tu-bs.de

present the hardware platform, as this is an integral part of development simplification in our view. The system is evaluated using a Wizard of Oz user study which supplies initial results for the impact of the `dinam` concept on developmental effort in an educational setting.

II. Related Work

There are numerous examples of mote-type sensor nodes [1]. Two examples of rather small, simplistic WSN development platforms are the uPart Sensor nodes [4] from the TecO group at Karlsruhe Institute of Technology (KIT) and the MITes [13] from House_n at Massachusetts Institute of Technology (MIT). Both have a single purpose (gathering sensory data), are as simplistic as the task allows, and both attempt to make the development and operation processes as easy to conduct as possible. Because both systems are intended to be configured and not to be programmed, application programming effort is shifted to the program running on a client, e.g. an end-user PC, and away from the WSN itself.

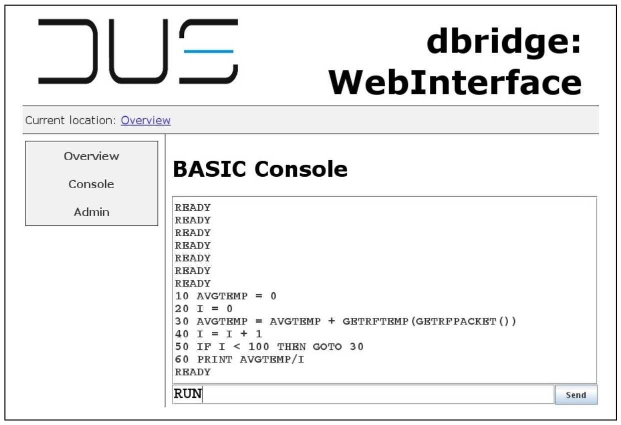
An example of a simplified application-building tool is the Arduino development platform for embedded electronics[2]. The Arduino platform is delivered with its own intuitive development environment which allows developers to create applications extremely quickly with only a basic knowledge of the ANSI C programming language. Similar to `dinam`, low-level functionality of the hardware architecture is wrapped using programming primitives in order to increase the ease of use for first time users and non-experts. But a separate development environment as well as distinct offline (development-time) and online (run-time) phases are not in accordance with our fully-integrated `dinam` concept. The mbed™ project from ARM and NXP [10] uses a browser-based development environment provided via an externally hosted decentralized server environment. While removing the installation time of the development environment is a step in the positive direction, the discrete development process is still an unnecessary hindrance.

[3] contains a discussion of archetypes for developing WSN programming languages to increase development efficiency. In [11] a BASIC interface for WSN development is presented which simplifies ease of use of the development interface. Still, discrete program, flash and run steps as well as IDE installation increase system complexity unnecessarily and open the door for incompatibilities and misunderstandings. ByVAC BASIC [12] for the PIC32 micro-controller is another approach aimed at easing development of embedded systems, but also requires a client-side development environment.

A previous step towards the `dinam` mote is the D-Bridge in [6] which implements a self-contained application in a wireless sensor network but lacks an integrated development environment. The platform greatly reduces installation overhead and was therefore selected as the basis for the further development of the `dinam` prototype.

III. Concept

In order to reach the goal that *simple applications must be simple to develop*, our system consists of



```
READY
READY
READY
READY
READY
READY
READY
10 AVGTEMP = 0
20 I = 0
30 AVGTEMP = AVGTEMP + GETRFTEMP(GETRFPACKET())
40 I = I + 1
50 IF I < 100 THEN GOTO 30
60 PRINT AVGTEMP/I
READY
```

```
RUN
```

Fig. 1. A Screenshot of the `dinam` Prototype Interface

- an integrated development environment.
- a (conceptual) sensor node that contains both the complete development environment and required data (programs, configurations, past versions).
- a network interface to the sensor node and its integrated development environment using Web/AJAX technology.
- a minimal sensor network system consisting of a bridge (for Internet to sensor network coupling) and sensor nodes that run off-the-shelf after power-on.

Development processes for embedded sensor systems addressing non-expert users are not completely novel, and `dinam` is inspired by the Arduino platform [2]. But our approach goes one step further: we simplify these components and integrate the development environment – including the data – into the wireless sensor system. This means, with `dinam` there is no development software required at the application programmer's computer to write programs for the sensor node – not even an editor is required. All code and the development environment is integrated into the node itself. Access to the node and the development environment is provided through a web-server integrated into the WSN system, thus the only tool a developer requires is a web-browser. Fig. 1 shows the developer's view of the IDE running in the WSN: the web-interface provides simplified access for easy operation. Because the development environment is integrated into the sensor system, hardware and software versions of all development components are always synchronized and incompatibilities do not occur. Our approach also removes some of the time consuming steps in the development process, such as writing the binary to the hardware, as well as reduces the initial static set-up time and effort overhead.

To simplify development, the `dinam` concept calls for integrating the discrete development steps into a single fluid process. Thus, the offline code generation time and online run-time must be combined. This will create a single constant run/development time. During this operational period, the developer must be able to receive immediate feedback to his/her actions so that the evaluation step can be pipelined as well. The resulting `dinam` system contains an interactive run-time programming interface which allows the programmer to change parameters, execute actions and evaluate the results

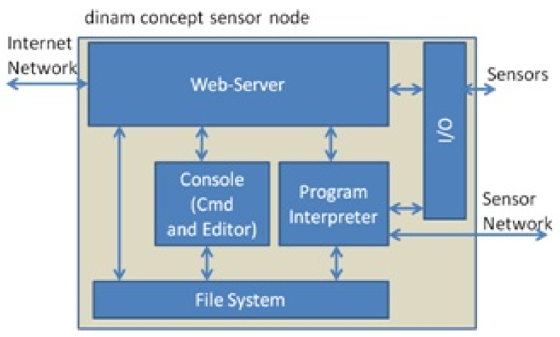


Fig. 2. `dinam` concept sensor node

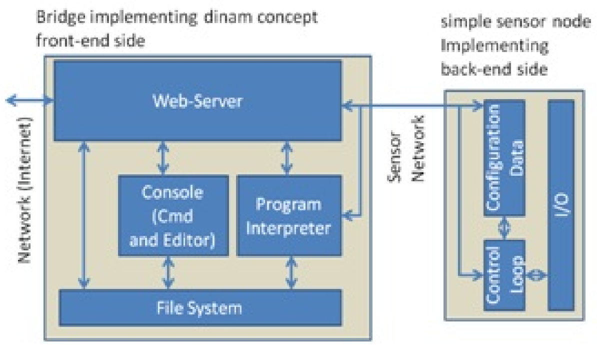


Fig. 3. `dinam` concept **split** sensor node with front end

simultaneously.

The command interface is opened by pointing the browser to the interface webpage located at the IP address which device acquires via DHCP. Once the page has been loaded, commands can be inputted directly to the interpreter using the Java-based web application shown in fig. 1, whereby the interpreter provides immediate evaluative feedback of previously issued commands. Our interpreter is inspired by the Commodore 64 BASIC [9] interface and interpreter with modified custom command words and syntax specifically for the `dinam` WSN application. Along with the BASIC command line interface, it also provides an editor to implement BASIC scripts. The interpreter associates one dedicated thread of execution with each individual script. Threads are concurrently executed, effectively resulting in interleaving execution semantics of the scripts. This development archetype combines all of the discrete development steps of mote-based development into a single continuous, fluid process.

The `dinam` concept is implemented on the `dinam` concept sensor node. The node consists of five components (fig. 2) to provide the described functionality. The *Web-Server* acts as the central access point to node functionality and status – e.g. status of I/O, configuration data or output produced by BASIC programs. Within the Web-Server, the *Console* offers access to the programming functionality. C64 BASIC was selected as programming language, as it was designed for non-expert programmers and also contains enough machine oriented programming constructs so as to allow interaction with a node on all levels of operation [8].

Programs are interpreted by the *Program Interpreter*, which has access to *I/O*, the emphFile System, the Internet and other nodes via the sensor network. The *File System* serves as a data store for the *Web-Server* (e.g. all web-pages) and programs which are running on the *Interpreter*. It also contains the program code, which can be loaded and started using a run command on the console. The *File System* stores system configurations as well as any other auxiliary data. The *I/O* component provides access to external sensors as well as typical I/O (e.g. ADC, DAC, PWM, I2C, SPI). A program may store the last temperature reading received (I/O) to the file system and/or send a message when the average temperature rises above a certain threshold.

Due to high memory and processing resources required by the `dinam` concept sensor node, it is difficult to implement using today's technology, especially when assuming very long sensor node lifetime requirements. Alternatively, the node functionality can be split into a part that is implemented on a proxy device, e.g. a bridge, and a part that is running on a simple, ultra-low-power sensor node (see fig. 3). This approach is achievable using today's technology, and was prototyped and evaluated for this paper.

IV. The Prototype

The implementation was done using a combination of Akiba wireless sensor nodes and the D-Bridge dynamic bridge [6] as can be seen in fig. 4. The Akiba node is based on the PIC18F14K22 8-bit micro-controller from Microchip, and was developed at the Technische Universität Braunschweig specifically for this application. The processor runs at up to 16 MIPS with 512 B RAM, 256 B EEPROM and 16 kB ROM. Wireless communication occurs at 2.4 GHz over the ChipCon CC2500 from Texas Instruments with a printed PCB antenna. The system is on a single-sided PCB measuring 20mm x 18mm with a CR2032 coin cell battery on the back. Due to low-power technology such as wake-on-radio, standby receive modes have consumption rates in the low double digit microwatt range with transmission and reception consumption rates between 10 and 30 mW. This yields a lifetime of many months, or even years, on a single coin-cell battery depending on the how often data is received or transmitted.

The Akiba sensor node is equipped with light, temperature and vibration sensors as well as connection points for up to 4 analog external sensors and sensor/processor boards. On top of the RF communication interface, the nodes implement an over-the-air-configuration protocol that allows for remote configuration of parameters such as duty cycle length and sensing rate comparable to those found in uParts [4]. The node is well adapted to post-hoc computing applications along the lines of Smart-Its [5] which make them specifically suitable for the `dinam` concept. The D-Bridge is a mini embedded web-server which serves a dynamic web application to Ethernet-based IPv4 networks. It also has a communication module which allows it to communicate with the Akiba network, implementing a WSN to Ethernet network bridge. The basic web application on the D-Bridge has been expanded to implement the front-end of the `dinam` concept sensor node.

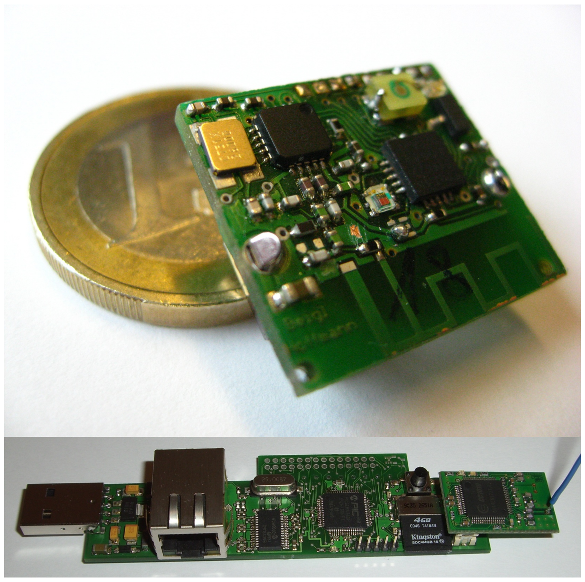


Fig. 4. D-Bridge and Akiba Wireless Sensor Node

V. Evaluation

In order to confirm or refute the claim that applications can be developed in under 10 minutes using the dinam prototype, a Wizard of Oz-based user study in an educational setting was designed to generate initial feedback as to the effectiveness of the concept. Each test subject was instructed on the use of the BASIC interface for a period of five minutes and then asked to complete a simple task using the dinam prototype.

The tutorial consisted of a BASIC syntax introduction as well as information about the interface and usage, including special functions applying to WSN functionality. Following the tutorial, each of the subjects was asked to implement an application which collected 100 temperature values transmitted by a sensor node and print the average of these values to the console. Both the sensory input and the console output were simulated in the study.

The time required by each of the subjects to complete the task was recorded. In total, five subjects were observed in the study. All five were students (three computer science, one computer engineering and one electrical engineering), had no prior knowledge of the tasks or experience with this, or any other BASIC dialect. The results ranged from 3:50 mins to 7:11 mins with an average time of 5:20 mins. Although no user studies were conducted on time required for traditional mote-based development approaches, in almost all cases the time required for step two and three (installation of the development environment and connecting the devices) is greater than the total development time using the dinam prototype.

During the course of the test, two phenomena were observed which may indicate some interesting aspects of the dinam prototype's development interface. First, test subjects who used the text editor field to construct programs where able to accomplish the task significantly faster than those using the console. This would seem to indicate that a console, while useful for debugging, is not an efficient way of creating applications. Second, students with more programming experience were able to accomplish the task faster than those with less experience. This is demonstrated by comparing the average result for computer science students (4:38 mins) to the average result for non-computer science students (5:33 mins). This would seem to indicate that although the learning curve has been drastically reduced, it is still dependent on the programming experience of the subject.

VI. Conclusion and Outlook

In this paper, the dinam concept has been presented. We showed that this concept is a different, if not orthogonal, concept to that of mote-based WSN development. The basic principle behind dinam is the integration of the execution platform and the development platform into one single conceptual device, and the use of the Web to access the development tools. We showed that this approach enables development of simple applications in under 10 minutes including an instruction period. The most complex portion of the dinam concept node, the basic interpreter, has been implemented on the D-Bridge, and is the subject of ongoing development and research. Furthermore, extensive user studies involving other application building archetypes, more users and real scenarios are ongoing to evaluate development time and cognitive load incurred by different approaches.

References

[1] Sensor network museum. http://www.snm.ethz.ch/Main/HomePage.

[2] Arduino: an Open-Source Electronics Prototyping Platform. http://www.arduino.cc/, 2010.

[3] Lan S. Bai, Robert P. Dick, and Peter A. Dinda. Archetype-based design: Sensor network programming for application experts, not just programming experts. In *IPSN '09: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 85–96, Washington, DC, USA, 2009. IEEE Computer Society.

[4] Michael Beigl, Albert Krohn, Till Riedel, Tobias Zimmer, Christian Decker, and Manabu Isomura. The upart experience: Building a wireless sensor network. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 366–373, New York, NY, USA, 2006. ACM.

[5] Hans Gellersen, Gerd Kortuem, Albrecht Schmidt, and Michael Beigl. Physical prototyping with smart-its. *IEEE Pervasive Computing*, 3(3):74–82, 2004.

[6] Dawud Gordon and Michael Beigl. D-bridge: A platform for developing low-cost wsn product solutions. In *Proceedings of the Sixth International Conference on Networked Sensing Systems (INSS09)*, pages 62–65, Pittsburgh, Pennsylvania, 2009. IEEE.

[7] Paul Holleis. *Integrating Usability Models into Pervasive Application Development*. PhD thesis, LMU, Munich, 2008.

[8] Caitlin Kelleher and Randy Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.*, 37(2):83–137, 2005.

[9] Thomas E. Kurtz. Basic. *History of programming languages I*, pages 515–537, 1981.

[10] Advanced RISC Machines Ltd. The MBED Rapid Prototyping Platform for Microcontrollers. www.mbed.org, 2010.

[11] J. Scott Miller, Peter A. Dinda, and Robert P. Dick. Evaluating a basic approach to sensor network node programming. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168, New York, NY, USA, 2009. ACM.

[12] ByVac PIC32-Basic. http://www.pic32.byvac.com/, 2010.

[13] Emmanuel Munguia Tapia, Stephen S. Intille, Louis Lopez, and Kent Larson. The design of a portable kit of wireless sensors for naturalistic data collection. In *Pervasive Computing, 4th International Conference, PERVASIVE 2006*, volume 3968 of *Lecture Notes in Computer Science*, pages 117–134, Dublin, Ireland, May 2006. LNCS.